INTRODUZIONE

A

ETHERNET

Benvenuti nel mondo delle LAN

[Introduzione](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/introduzione.html" \l "introduzione)

[Ethernet e Token Ring: i due tipi più diffusi](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/introduzione.html" \l "ethernet-token)

INTRODUZIONE

Se state leggendo questa introduzione, probabilmente siete a digiuno di reti. Cercherò quindi di introdurvi a questo argomento e contemporaneamente a questo sito.

Una LAN, Local Area Network, cioè rete locale, è un sistema informatico costituito da più computer, comunicanti attraverso un qualche sistema trasmissivo. La rete è locale perchè la sua estensione non è molto vasta, al massimo non supera i 2Km, e spesso non si estende per più di 500m. . Si differenzia per questo dalle MAN, Metropolitan Area Network, con estensioni paragonabili a quelle di una città, e dalle WAN, Wide Area Network, reti di vaste dimensioni a livello geografico.

Attraverso una rete non solo è possibile scambiare file o messaggi, ma anche diversi altri servizi, quali le stampanti o i modem, collegati fisicamente ad un solo nodo della rete, ma accessibili anche da tutti gli altri computer connessi.

ETHERNET e TOKEN RING: i due tipi più diffusi

Scendendo più nel dettaglio, vediamo che sono due i tipi di rete maggiormente diffusi: Ethernet, di cui parlerò più specificatamente nel sito, e token ring di IBM. La differenza principale tra questi due tipi è che, mentre Ethernet possiede una struttura elettrica a bus, token ring è ad anello.

Un altra differenza molto importante è che, mentre in una rete Ethernet l'accesso alla rete è regolato dal sistema [CSMA/CD](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html" \l "csmacd), una tecnica che permette ad un computer di usare la rete solo se nessun altro la stava già usando, token ring sfrutta la sua struttura anulare facendo passare, ad intervalli predefiniti, da un computer all'altro, una "moneta virtuale"; solo chi è in possesso di tale moneta può trasmettere. La conseguenza immediata di questa diversità è quale tipo di traffico la rete riesce meglio a gestire. Per chiarire meglio le cose facciamo un esempio.

Immaginate di essere collegati in video-conferenza con un'altra persona usando una rete Ethernet. Tutto funziona benissimo finchè, ad un certo punto, un altro computer si impadronisce della rete sfruttando una brevissima pausa nella vostra comunicazione, supponiamo anche che tale computer abbia intenzione di trasferire un file grande diverse decina di Mb. Cosa succede?

Succede che la vostra video-conferenza si interrompe bruscamente. L'unica cosa che si può fare è, o attendere pazientemente che il trasferimento di quel file finisca, oppure, se siete fortunati, riprendere il controllo della rete sfruttando una micro pausa generata da quel computer "prepotente" (basta, per esempio, il ritardo creato dal riposizionamento di una testina del disco rigido di quel computer). In ogni caso la situazione potrebbe ripresentarsi dall'inizio, e il flusso audio-video tra voi e il vostro interlocutore verrebbe trasmesso a scatti (voce compresa).

Se invece di una rete Ethernet, usaste una token ring, le cose migliorerebbero. Infatti, anche se un altro computer iniziasse a trasferire un file molto lungo, grazie alla politica della moneta, ogni computer avrebbe a disposizione un intervallo di tempo fisso per trasmettere i dati, allo scadere del quale la moneta passa al computer successivo. E'questo il motivo per cui si dice che token ring è una rete di tipo deterministico. In questo nuovo scenario, perchè la vostra video-conferenza proceda senza interruzioni, sarà sufficiente un semplice buffer (una memoria cuscinetto). Esso farà fronte alla pausa generata nel periodo di tempo in cui non avete la moneta e state aspettando che vi ritorni in possesso.

Sembrerebbe quindi che Ethernet sia un tipo di rete inefficiente, almeno rispetto a token ring. Perchè allora Ethernet è così diffusa?

Le ragioni del suo successo sono essenzialmente la sua semplicità e la sua economicità. Token ring infatti, è una rete che richiede maggiori costi in fase di installazione, oltre che un maggiore onere di gestione.

Il viaggio dei dati

[Il modello ISO/OSI](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html" \l "iso-osi)

[Imbustamento multiplo](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html" \l "imbustamento)

[I sette layers in dettaglio](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html" \l "layers)

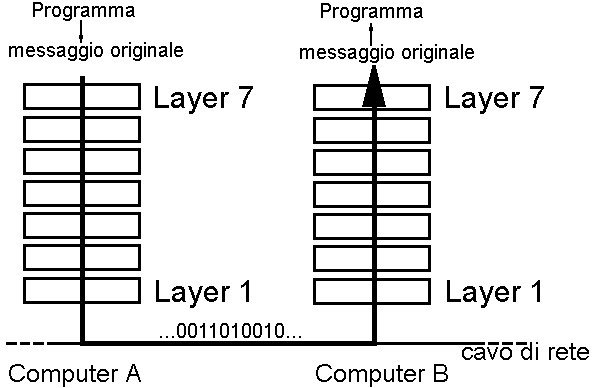
[Relazione tra livelli adiacenti](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html" \l "relazione-livelli)

Un'operazione semplice come trasferire un file da un computer ad un altro, nasconde al suo interno una serie di processi che meritano un approfondimento.

IL MODELLO ISO/OSI

Lo standard che un pò tutte le reti locali seguono è l'OSI (Open System Interface) definito negli anni 80 da un ente, l'ISO: Organizzazione Internazionale per la Standardizzazione. L'OSI è un modello di riferimento che divide l'operazione da cui siamo partiti in sette problemi più piccoli e più facili da analizzare, detti layer (livelli).

Vediamo cosa succede ai nostri dati mentre passano da un layer all'altro. Supponiamo di volere mandare un messaggio da un computer A ad un altro computer B. Ecco cosa succede



Come si vede nella figura qui sopra, l'applicazione del computer A genera il messaggio. Poi questo scende nel primo layer, il 7, poi nel 6 nel 5 e così via fino a che non arriva al layer 1.

IMBUSTAMENTO MULTIPLO

Durante ogni passaggio, intorno ad esso, vengono via via aggiunte informazioni supplementari, in modo tale che esso sia pronto ad affrontare il suo lungo viaggio attraverso terre potenzialmente ostili. Il nome con cui ci si riferisce a suddetti dati, è:

header (se essi sono aggiunti in testa)

oppure

trailer (se aggiunti in coda).

Perchè tutto proceda per il verso giusto questi dati aggiuntivi sono molto utili. I loro compiti sono sostanzialmente i seguenti:

il messaggio deve essere recapitato al computer giusto

non deve contenere errori o alterazioni

deve seguire certi formati, definiti dai protocolli utilizzati

Quello che succede è che il messaggio viene rielaborato dal layer corrente in modo tale da essere "digerito" dal layer che segue, ed i dati extra non sono altro che il "condimento". Una volta aggiunti, il messaggio diventa di livello più basso.

Questa tecnica è chiamata imbustamento multiplo, il nome si spega bene da solo.

Una volta giunto fino al computer B, il messaggio risale i sette layer seguendo la procedura inversa.

In questa seconda fase ogni layer esamina ed usa le informazioni aggiuntive. Se tutto procede senza intoppi, il messaggio viene spacchettato, cioè viene a poco a poco ripulito da questi dati addizionali, finchè non si sarà ripristinato il messaggio originale da visualizzare sul programma destinatario.

Non è detto che un'architettura di rete implementi in modo esplicito tutti questi sette layer, spesso è meglio riunirne qualcuno in un blocco unico.

Approfondimento: i 7 layer del modello OSI visti da vicino

LAYER 1: FISICO (il più vicino all'hardware)

Il livello 1 si occupa della trasmissione dei dati in binario sulla rete. In questo livello viene specificata la rappresentazione elettrica degli 0 ed 1 binari in modo tale che il sistema ricevente possa leggere correttamente il flusso di bit in arrivo.

LAYER 2: DATA LINK

Il livello 2 serve per gestire le trame (frame), spezzoni di messaggio di lunghezza fissa. Gestisce inoltre meccanismi per l'individuazione e la correzione degli errori avvenuti nel layer 1.

LAYER 3: NETWORK

Il livello 3 gestisce l'instradamento dei pacchetti, esaminandone l'indirizzo di provenienza e di destinazione e gestendo le tabelle di instradamento necessarie per giungere velocemente a destinazione. E' anche usato per scegliere instradamenti alternativi nel caso di guasti o di mutate condizioni di traffico su certi segmenti. I router lavorano su questo layer.

LAYER 4: TRASPORTO

Il livello 4 si occupa di fornire un trasferimento dati affidabile. In particolare partiziona i messaggi in pacchetti, ne controlla e preserva l'ordine con cui devono essere spediti, controlla la presenza di errori. E' il primo layer che lavora indipendentemente dalla topologia della rete in cui si trova, per questo viene chiamato end-to-end.

LAYER 5: SESSIONE

Il livello 5 controlla il dialogo e la sincronizzazione tra due programmi applicativi, entità del layer presentazione, il 6. Tra i suoi servizi ci sono inoltre le comunicazioni in remoto, ed il trasferimento di posta e file.

LAYER 6: PRESENTAZIONE

Il livello 6 serve per risolvere le differenze di formato dei dati. Questo può accadere se in rete ci sono sistemi che adottano codifiche diverse (ad esempio codifica ASCII o EBCDIC).

LAYER 7: APPLICAZIONE (il più vicino al software)

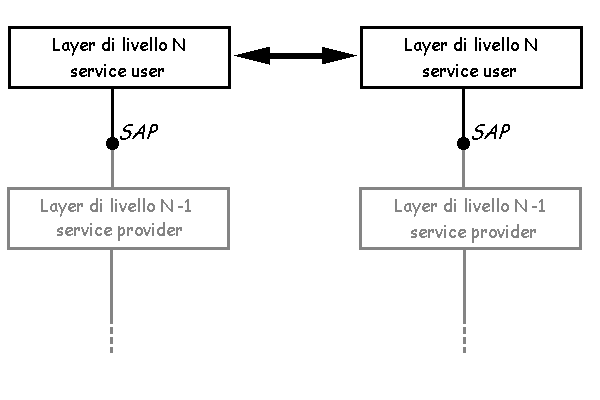
Il livello 7 è il layer che interagisce direttamente con i programmi applicativi sul sistema operativo. Si occupa tra l'altro del trasferimento di file, delle connessioni a computer remoti e della posta elettronica.

RELAZIONE TRA LIVELLI ADIACENTI

Come abbiamo visto, ognuno di questi sette layer comunica direttamente con il layer sottostante.

Tuttavia, guardiamo per un momento la struttura OSI da un altro punto di vista. E' abbastanza facile rendersi conto che in realtà, se prendiamo un layer di livello N su un computer A, esso vuole parlare non tanto con il suo layer sottostante di livello N-1 , bensì con il layer dello stesso livello N, ma del computer B. E' come se tutto quello che sta sotto i due layer di livello N sia solo un servizio perchè essi comunichino.

Un layer di livello N viene chiamato service user, uno di livello N-1 è detto service provider. I servizi di N-1 sono forniti ad N in un punto detto SAP, service access point, che è semplicemente il luogo in cui N può chiedere i servizi di N-1.



Il ragionamento si può ripetere per ogni layer, escludendo il primo che trasmette direttamente i suoi dati sulla rete fisica.

In che modo? ve lo spiego nella [sezione relativa](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html).

Come i dati vengono spediti sul cavo

[La codifica Manchester](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html" \l "manchester)

[La ricezione](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html" \l "ricezione)

[La trasmissione - CSMA/CD](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html" \l "csmacd)

[Le collisioni](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/csmacd.html" \l "collisioni)

LA CODIFICA MANCHESTER

Un messaggio, quando viaggia sul cavo, non è altro che una lunga sequenza di bit, che possono essere 0 o 1. Esso raggiunge tutti i nodi collegati alla rete, indipendentemente da dove sia fisicamente posizionato il destinatario.

I valori binari 0 e 1 sono rappresentati tramite la codifica Manchester. Essa presuppone che venga generata un onda quadra, la portante, che oscilla tra due valori di tensione predefiniti, uno positivo e l'altro negativo. Nel momento in cui l'onda cambia di segno è presente un 1 (da - a +) o uno 0 (da + a -).

LA RICEZIONE

Prima di iniziare ricordo che, oltre ai dati veri e propri, il messaggio contiene al suo interno anche tante altre informazioni, tra cui troviamo l'indirizzo di destinazione.

In qualsiasi rete Ethernet, tutti i computer collegati rimango sempre in ascolto. Quando qualcuno manda un messaggio in rete, tutti gli altri computer, o meglio, tutte le schede di rete, iniziano ad analizzare il messaggio in transito.

La prima cosa che effettua una scheda di rete, è la lettura dell'indirizzo di destinazione, solitamente posizionato in posizione strategica in testa al messaggio. Solo se questo coincide con il proprio, il messaggio viene accettato e letto, altrimenti verrà scartato.

Esistono anche alcuni messaggi, detti di tipo broadcast, che vengono accettati da tutte le schede di rete in ascolto.

LA TRASMISSIONE - CSMA/CD

Il metodo per disciplinare l'accesso alla rete è il CSMA/CD, Carrier Sense Multiple Access/Collision Detection, accesso multiplo con rilevamento della portante/rilevamento di collisione. Ogni scheda di rete segue questa tecnica quando deve spedire un messaggio.

Come è ragionevole pensare, la prima cosa da fare quando si vuole trasmettere, è verificare che la rete non sia già stata occupata da qualcun'altro, ossia non deve essere rilevata l'onda portante di cui abbiamo parlato prima.

LE COLLISIONI

Questo controllo, purtroppo, non assicura che solo un computer alla volta inizi a trasmettere. Non è infatti infrequente che due o più computer, trovando la rete libera, inizino a trasmettere contemporaneamente. Inoltre bisogna considerare che i dati non raggiungono ogni estremità della rete in tempo nullo, ma viaggiano sul cavo con una certa velocità di propagazione. Basta quindi che due computer, fisicamente distanti qualche centinaio di metri, inizino a trasmettere in due istanti sufficientemente vicini perchè si ricrei la medesima situazione. In questi casi si ha una collisione.

Questo evento non è così catastrofico come si potrebbe pensare. La scheda di rete, infatti, anche quando è in trasmissione, rimane costantemente in ascolto. Quando avviene una collisione, le onde generate si sommano, il risultato è che le oscillazioni dell'onda risultante variano tra intervalli di tensioni maggiori di quelli predefiniti. In questa situazione le schede coinvolte rilevano l' "onda anomala" e terminano di trasmettere, non subito, ma solo quando hanno inviato almeno 64 byte, in modo tale da permettere che anche tutte le altre schede di rete rilevino la collisione. Terminata la trasmissione, viene impostato un timer a durata casuale nella scheda di rete, allo scadere del quale si ritenta la comunicazione.

Analisi delle trame Ethernet

In una rete Ethernet, quando si vogliono inviare dei dati, bisogna prima di tutto costruire un pacchetto atto a contenerli. Questa struttura conterrà, incapsulati, i dati originali; intorno a questi verranno aggiunte varie informazioni, utili perchè i dati arrivino al computer giusto e privi di errori. Tale pacchetto viene chiamato frame, oppure trama. Vedremo che ne esistono di quattro tipi.

Il primo modello di trama è stato definito nei primi anni 80 dallo standard Ethernet II. Ha una struttura molto semplice. Contiene i seguenti campi

* Preambolo; 7 byte contenenti una sequenza di 101010... , serve per sincronizzare tutte le stazioni in ascolto, con il segnale generato dalla scheda di rete.
* SFD, Start Frame Delimiter; un byte contenente la sequenza 10101011. Con gli 101010 si accoda al preambolo, con l' 11 finale avverte che stanno per arrivare i dati veri e propri.
* Indirizzo di Destinazione; 48 bit (6 byte), contiene l'indirizzo fisico del nodo destinatario.
* Indirizzo di Provenienza; 48 bit (6 byte), contiene l'indirizzo fisico del nodo sorgente. Questi ultimi due indirizzi sono lunghi proprio, guardacaso, 48 bit.
* Tipo; 2 byte, contiene un valore che indica in quale ambiente è stato generato il pacchetto.
* Dati; da 46 a 1500 byte. Sono i dati veri e propri. Se sono più lunghi di 1500 byte, si costruiranno altri pacchetti da 1500 byte, finchè non viene spedito tutto.
* FCS, Frame Check Sequence; contiene il risultato del calcolo CRC sul campo dati. Serve al destinatario, per controllare se i dati hanno subito qualche alterazione.

In Ethernet, se qualcosa nella trasmissione va storto, il computer ricevente scarta il messaggio, ma voi non potete saperlo. In genere sono le applicazioni di alto livello che controllano il buon esito di un operazione.

La trama Ethernet II supporta TCP/IP ed IPX/SPX (NetWare).

La seconda trama creata dopo Ethernet II, è Ethernet 802.3. Si tratta di una trama proprietaria, usata dalle stazioni Novell Netware 3.x, e supporta solo IPX/SPX. I campi sono molto simili, cambia solo il campo "tipo", che resta lungo 2 byte, ma ora viene usato per contenere la lunghezza della trama.

Alla fine degli anni 80, venne creata una terza trama: Ethernet 802.2. Si differenzia dalla 802.3 per l'aggiunta di 3 campi da 1 byte, sottratti al campo dati (che diventa lungo al massimo 1497 byte). I tre campi sono

* DSAP, Destination Service Access Point; indica a quale processo di alto livello bisogna consegnare la trama.
* SSAP, Source Service Access Point; indica il processo software di provenienza.
* Controllo; indica il tipo di trama, secondo uno standard definito dall' IEEE.

La 802.2 è la trama ufficiale di Netware 4.x, è in grado di gestire qualunque protocollo.

Una quarta ultima variante della trama Ethernet è la SNAP. E' simile alla 802.2, ma vengono aggiunti due nuovi campi, sottraendo altri 5 byte al campo dati (che si riduce ad un massimo di 1492 byte). Il primo campo è di 3 byte e contiene un codice identificativo del fornitore, negli altri 2 byte ritroviamo lo stesso campo Tipo usato nella Ethernet II.

Il protocollo

TCP/IP

[Introduzione a TCP/IP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "introduzione)

[Il protocollo TCP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "tcp)  
- [L' header TCP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "headerTCP)  
- [Il meccanismo delle finestre scorrevoli](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "finestre)  
- [Stabilire una connessione TCP: l'handshake a tre vie](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "3wayhandshake)  
- [Chiudere una connessione TCP: il saluto a quattro vie](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "saluto4vie)  
- [Gli stati di una connessione TCP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "statiTCP)  
- [Il controllo delle congestioni di TCP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "congestioni)

[Il protocollo UDP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "udp)

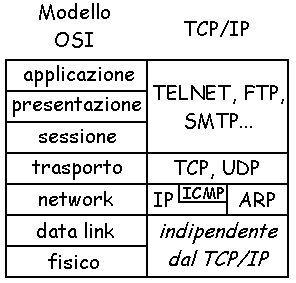
[Il protocollo ICMP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "ICMP)

[Il protocollo IP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "ip)  
- [L'header IP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "headerIP)  
- [La frammentazione dei datagrammi IP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "frammentazione)  
- [Indirizzi IP - Indirizzi Ethernet](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "ip-ethernet)

[Il protocollo ARP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/tcpip.html" \l "arp)

INTRODUZIONE A TCP/IP

Innanzi tutto spieghiamo il nome. TCP sta per Transfer Control Protocol, IP sta per Internet Protocol. In realtà il protocollo TCP/IP è una famiglia di protocolli, ognuno con una sua funzione particolare, TCP ed IP sono due protocolli importanti di questa famiglia.



Alcuni di questi, come TCP, UDP, IP, ICMP e ARP (li vedremo meglio dopo) sono di *basso livello*. Questo significa che essi lavorano vicino al livello fisico della rete (vedremo infatti che i loro compiti sono strettamente correlati con la trasmissione effettiva dei dati). La loro funzione è di fornire servizi ai *protocolli superiori* e alle applicazioni. Se non avete capito bene questo concetto vi consiglio di leggere la pagina [Il viaggio dei dati](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html).

Tutti i protocolli di livello superiore, sono specializzati nel compiere qualche servizio particolare. Ad esempio FTP, File Transfer Protocol, permette il trasferimento dei file, TELNET, network terminal protocol, permette di parlare con un computer remoto come se vi foste davanti, SMTP, Simple Mail Trasfer Protocol, serve a trasferire la posta elettronica, SNMP, Simple Network Management Protocol, per gestire e monitorare lo stato della rete. Questi non sono tutti, sono solo i più importanti. Per la loro diversità, ritengo che descriverli uno per uno in dettaglio, andrebbe al di là dello scopo di questo documento.

I protocolli di livello superiore basano il loro lavoro su un'assunzione: che i protocolli di basso livello siano in grado di comunicare in modo affidabile.

IL PROTOCOLLO TCP

Vediamo questa struttura al lavoro facendo un semplice esempio, immaginate di voler inviare un messaggio su una rete TCP/IP (Internet è una di queste). Ciò che il protocollo di posta SMTP produce, viene passato in basso al TCP.

Innanzi tutto diciamo che TCP fornisce un servizio byte-stream. Questo termine indica che i dati da e per i livelli superiori vengono presentati e ricevuti come un unico flusso di byte, e non come pacchetti. Questo significa che è TCP, e non i protocolli superiori (e quindi le applicazioni), a doversi preoccupare di preparare i pacchetti, con l'evidente vantaggio di avere una chiara la distinzione dei ruoli: SMTP (o qualsiasi altra applicazione di alto livello) pensa a preparare il messaggio di posta, TCP pensa a come deve inviarlo.

Un pacchetto TCP è detto segmento.

Essenzialmente i compiti di TCP sono:

* suddividere i dati da spedire in tanti segmenti indipendenti e numerati.
* riassemblare i dati arrivati all'altro capo, presentandoli nuovamente come un flusso di byte.
* rispedire i datagrammi non arrivati o arrivati corrotti.
* rimetterli nel giusto ordine se alcuni di essi non hanno "rispettato il turno".
* controllare il flusso attraverso il meccanismo delle finestre e dell'acknowledgement che descrivo tra poco.
* multiplexing attraverso l'uso delle porte. Ne parlo nell'header TCP.

L'header TCP

Ad ogni segmento, TCP aggiunge il suo header. Se non sapete cos'è un header dovreste davvero leggere la pagina [Il viaggio dei dati](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html).

Nell'header sono contenuti i seguenti campi; quelli in rosso sono i più importanti

* Source port/Destination port; (16 bit + 16 bit) quando iniziate ad inviare o a ricevere dati, qui vengono memorizzati due numeri di porta, uno al quale inviare i dati, uno sul quale i dati verranno spediti da parte dell'altro computer. Sull'altra macchina, i numeri di porta sono uguali ma scambiati. Per ogni comunicazione full-duplex, sul vostro computer (e su quello a cui vi siete connessi) vengono aperte due connessioni, una in trasmissione e una in ricezione, ognuna con il suo numero di porta. Fate attenzione a non confondere le porte con gli indirizzi IP.

Qui apro una piccola ma importante parentesi per far capire quanto sia utile il concetto di porta nelle comunicazioni TCP. Normalmente, quando stabilite una comunicazione generica, TCP genera un numero di porta casuale. Se però volete iniziare una connessione con un protocollo di alto livello (cioè il 99% dei casi), come per esempio il protocollo SMTP, perchè volete mandare un messaggio di posta all'altro computer, dovete prima di tutto specificare che volete una connessione attraverso la porta n.25 (o lo farà per voi il vostro lettore di posta). Solo dietro a questo numero di porta troverete SMTP in ascolto. Ci sono diversi "numeri magici" standard (detti "well known services", servizi ben noti), ad esempio 21 per FTP, 23 per Telnet, 53 per il servizio DNS, 80 per il Web Server. Potete trovare una lista dei principali servizi noti in [C:\Windows\Services](C:\\Windows\\Services), o in /etc/services in Linux.

* Sequence number; (32 bit) contiene il numero necessario per sapere quale sia l'ordine dei segmenti e per sapere se qualcuno è andato perduto. Diversamente da come si potrebbe pensare, non viene usato il numero di segmento, ma il numero del primo byte di quel segmento. Quindi se ogni segmento contiene 1500 byte, il sequence number sarà 0...1500...3000... , e non 0...1...2... .
* Acknowledgment number; (32 bit) viene usato per segnalare che avete ricevuto tutti i dati fino al numero di byte specificato meno uno, e dovrebbe essere uguale al valore del prossimo Sequence number che sarà ricevuto.
* Window; (16 bit) nei segmenti con il flag ACK, indica in byte l'ampiezza della finestra che il computer è in grado di ricevere. Per il funzionamento vedi il paragrafo Il meccanismo delle finestre scorrevoli.
* Data offset; (4 bit) è il numero di parole a 32-bit dell' header TCP. Indica dove iniziano i dati.
* Reserved; (6 bit) riservato ad usi futuri.
* Flags; (1 bit per ogni flag) è composto dai seguenti campi:
  + Flag URG; (urgent) se attivo indica che il campo Urgent Pointer è valido e deve essere letto.
  + Flag ACK; (acknowledgement) se attivo indica che il campo Acknowledgement number è valido e deve essere letto.
  + Flag PSH; (push) usato per indicare che il segmento deve essere inviato immediatamente. Utile nei programmi interattivi come Telnet, dove vogliamo che i comandi digitati con la tastiera siano inoltrati quanto prima. Senza il Flag PSH, per motivi di efficienza il TCP accumula tutti i dati in un buffer interno, la cui spedizione viene ritardata fino a che non si sia completamente riempito.
  + Flag RST; (reset) usato per reinizializzare completamente la connessione TCP corrente.
  + Flag SYN; (synchronize) usato quando viene stabilita una sessione, indica che il ricevente dovrà leggere il campo Sequence number e sincronizzare il proprio con esso.
  + Flag FIN; (finish) indica che non ci sono altri dati da trasmettere. La connessione rimane comunque aperta in ricezione.
* Checksum; (16 bit) un numero che serve per sapere se il datagramma corrente contiene errori nel campo dati.
* Urgent pointer; (16 bit) indica che il ricevente deve iniziare a leggere il campo dati a partire dal numero di byte specificato. Viene usato se si inviano comandi che danno inizio ad eventi asincroni "urgenti". Ad esempio il comando Control-C in una sessione Telnet.
* Options; (lunghezza variabile) contiene varie opzioni di TCP (Maximum Segment Size, Window Scale, Sack Permitted, Sack, Time Stamp).
* Padding; (lunghezza variabile) una serie di 0 inserita perchè la lunghezza dell'header TCP sia un multiplo di 32 bit.

E finalmente

* Data; il nostro campo dati.

Il meccanismo delle finestre scorrevoli

Cosi come l'abbiamo descritto potremmo pensare che il lavoro del TCP consista semplicemente nel mandare un segmento, aspettare l'acknowledgement per quel pacchetto, mandare un altro pacchetto. Se l'Ack non arriva entro un certo intervallo di tempo ritrasmettere il primo segmento. Anche se funziona, questo modo di procedere è estremamente inefficiente poichè sfrutta solo una piccola parte della larghezza di banda disponibile. Un sistema molto più efficiente ed ugualmente affidabile, è il meccanismo a finestre. Il funzionamento è il seguente.

* Il mittente invia una "finestra" di pacchetti TCP, da 1 a n, ma senza aspettare l'Ack, inoltre fa partire un timer per ognuno di essi. Il numero n è detto ampiezza della finestra.
* Il ricevente manda un Ack per ogni pacchetto.
* Il mittente sposta la finestra "in avanti" di un pacchetto per ogni Ack ricevuto nell'ordine. Cioè quando arriva l'Ack del segmento 1, la finestra viene spostata in modo da coprire i pacchetti da 2 a n+1 e viene trasmesso il segmento n+1.

Consideriamo ora qualche caso particolare.

Se il pacchetto 2 non arrivasse a destinazione, la finestra non verrebbe spostata oltre il pacchetto 1. Il destinatario manderebbe gli Ack dei pacchetti 3, 4, 5... ma tutti uguali, cioè settati al valore 1, dato che è il questo l'ultimo pacchetto valido, ricevuto nell'ordine di consegna. Ad un certo punto il timer per 2 scade e il pacchetto viene ritrasmesso. A questo punto però sorge una domanda: dobbiamo ritrasmettere anche 3, 4, 5... ? Purtroppo non possiamo saperlo. Se mandiamo solo 2, ma anche 3, 4, 5... sono andati persi dovremo aspettare che scadano i timer di tutti questi altri segmenti. Alternativamente possiamo rimandare tutta la finestra. E' comunque chiaro che nessuna soluzione è priva di inefficienze, perchè l'informazione del campo Ack non è sufficientemente espressiva: non dice nulla del frame ricevuto, dice solo qual'è l'ultimo frame valido ricevuto nell'ordine di consegna.

Altra caso particolare: il pacchetto 2 viene ricevuto correttamente, ma è l'Ack che viene perso. Semplicemente, il mittente riceverà prima o poi un Ack con valore 3, questo indica che tutti i pacchetti fino al terzo sono arrivati a destinazione, quindi anche il secondo. Dopo l'Ack 3 il mittente può spostare la finestra in avanti di 2 passi in una volta. La finestra ora coprirà i pacchetti da 4 a n+3.

Nella realtà, per identificare i segmenti si usa il Sequence number non il numero di pacchetto (vedi la def. di Sequence number, nell'header TCP). Inoltre l'ampiezza della finestra è variabile da parte del ricevente durante la connessione grazie al campo Window.

Stabilire una connessione TCP: l'handshake a tre vie

Quando due computer utilizzano TCP devono innanzitutto creare una sessione. La procedura attraverso la quale la sessione viene stabilita si chiama "three way handshacking", o handshacking a tre vie.

Se il computer client PC1 vuole connettersi al computer server PC2, succede questo:

1 - PC1 manda a PC2 un segmento TCP attivando il flag SYN.

2 - PC2 risponde a PC1 con i flag SYN e ACK attivi.

3 - PC1 risponde a sua volta con il flag ACK.

Al primo passo, PC1 attiva nel primo segmento il flag SYN per indicare a PC2 che il campo Sequence number è valido e che quindi deve essere letto. Il valore settato da PC1 in quel campo è detto Sequence number iniziale, o ISN (Initial Sequence Number). PC2 risponde attivando il flag SYN ed indicando un proprio ISN, inoltre attiva l'ACK indicando l'ISN+1 di PC1. Infine PC1 attiva l'ACK indicando l'ISN+1 di PC2. A questo punto la comunicazione è stabilita e PC1 può iniziare ad inviare gli altri segmenti.

Ad esempio:

n.\_\_\_mitt.\_\_\_dest.\_\_\_\_\_SYN Flag\_\_\_ACK Flag\_\_\_\_\_Seq. n.\_\_\_\_\_\_\_\_\_\_Ack. n.

1\_\_\_\_PC1\_\_\_\_PC2\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_No\_\_\_\_\_\_\_\_\_2481573249\_\_\_\_\_1875913495   
2\_\_\_\_PC2\_\_\_\_PC1\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_408548955\_\_\_\_\_\_2481573250   
3\_\_\_\_PC1\_\_\_\_PC2\_\_\_\_\_No\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_2481573250\_\_\_\_\_408548956

Un problema di questa tecnica è la vulnerabilità ad un famoso tipo di DoS (Denial of Service): il SYN-Flooding. In sostanza il client (PC1) richiede decine e decine di connessioni al server vittima (PC2) inondandolo di pacchetti SYN (inondare = to flood). PC2 risponde con tanti SYN-ACK, a cui però PC1 volutamente non risponde (in teoria dovrebbe completare gli handshake con un ACK per ogni connessione). PC2 dopo un pò smette di aspettare, ma quando i SYN ricevuti sono troppi, rimane pressochè paralizzato: dato che esiste un limite massimo al numero di connessioni che si possono stabilire su PC2, un eventuale utente PC3 che si collega a PC2 non verrebbe accettato perchè tutti i canali disponibili sono impegnati (ad aspettare ACK che non arrivano).

Chiudere una connessione TCP: il saluto a quattro vie

Quando si vuole terminare una connessione, si usa una procedura di terminazione a quattro vie. Un computer manda all'altro un segmento con il flag FIN attivo. Il flag FIN chiude la connessione in una sola direzione. All'altro capo verrà spedito tutto ciò che deve ancora essere spedito, dopodichè anche questo invierà un segmento con il flag FIN. Ora che la comunicazione è chiusa in entrambi i sensi, la sessione può considerarsi terminata..

Ad esempio:

n.\_\_\_mitt.\_\_\_\_dest.\_\_\_\_\_FIN Flag\_\_\_ACK Flag\_\_\_\_Seq. n.\_\_\_\_\_\_\_\_\_\_Ack. n.

1\_\_\_\_PC1\_\_\_\_\_PC2\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_408549508\_\_\_\_\_\_2481575301   
2\_\_\_\_PC2\_\_\_\_\_PC1\_\_\_\_\_No\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_2481575301\_\_\_\_\_408549509   
3\_\_\_\_PC2\_\_\_\_\_PC1\_\_\_\_\_Si\_\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_2481575301\_\_\_\_\_408549509   
4\_\_\_\_PC1\_\_\_\_\_PC2\_\_\_\_\_No\_\_\_\_\_\_\_\_Si\_\_\_\_\_\_\_\_408549509\_\_\_\_\_\_2481575302

Gli stati di una connessione TCP

Una sessione TCP può essere in diversi stati. Potete visualizzare lo stato di tutte le connessioni in corso sul vostro computer con il comando:  
netstat -na

Gli stati possibili sono i seguenti

* LISTEN: in attesa che qualcuno richieda una connessione
* SYN-SENT: durante la creazione di una connessione, indica che è stato inviato un segmento con flag SYN attivo e si sta aspettando il segmento di risposta (quello con i flag SYN/ACK).
* SYN-RECEIVED: durante la creazione di una connessione, indica che è stato ricevuto un segmento SYN, è stato inviato in risposta il SYN/ACK e che ora si sta attendendo l'ACK che completa l'handshake.
* ESTABLISHED: stato raggiunto dopo che l'handshake è stato completato con successo. La connessione è ora aperta e si possono trasferire dati.
* FIN-WAIT1: in attesa di una richiesta di terminazione della sessione da parte del computer remoto o di un acknowledgement della richiesta di terminazione della connessione precedentemente stabilita.
* FIN-WAIT2: in attesa di una richiesta di terminazione della sessione da parte del computer remoto.
* CLOSE-WAIT: in attesa di una richiesta di terminazione della sessione da parte del computer locale.
* CLOSING: in attesa dell'acknowledgement alla richiesta di terminazione da parte del computer remoto.
* LAST-ACK: in attesa dell'acknowledgement della richiesta di terminazione della connessione che è stata precedentemente inviata al computer remoto, include un acknowledgement della sua richiesta di terminazione della connessione
* TIME-WAIT: in attesa che passi abbastanza tempo in modo da essere sicuri che il computer remoto abbia ricevuto l'acknowledgement della sua richiesta di terminazione della connessione.
* CLOSED: quando la connessione è del tutto terminata.

Il controllo delle congestioni di TCP

Per evitare congestioni indesiderate sulla rete, TCP dispone di alcuni algoritmi di controllo.

**Slow start** (partenza lenta): quando due computer iniziano a comunicare, questo algoritmo impedisce al mittente di partire subito con una finestra di massima ampiezza (suggerita nel campo window del ricevente, detto anche "advertised window"). Questo meccanismo è molto utile quando la rete che unisce due computer non è una LAN diretta, ma una serie di collegamenti lenti (router e collegamenti WAN). Quando si stabilisce la comunicazione, abbiamo una finestra nel TCP del mittente detta congestion window, cioè finestra di congestione. All'inizio è fatta di un solo segmento, poi cresce esponenzialmente (2, 4, 8, 16...) mano a mano che arrivano gli ACK dal computer ricevente, finchè non si raggiunge il valore del campo window del ricevente. Spesso però, prima di raggiungere il valore indicato dall'advertised window, si arriva ad punto in cui alcuni pacchetti iniziano ad essere scartati dai collegamenti lenti intermedi. Questo indica che è opportuno fermare la crescita della finestra di congestione. Attenzione a non confondere la congestion window con l'advertised window, quest'ultima è usato solo dal ricevente per indicare quanti segmenti per finestra esso è in grado di accettare nel buffer di ricezione, mentre la congestion window è gestita dal mittente in base al numero di ACK giunti indietro.

**Congestion Avoidance** (evitare la congestione): Congestion Avoidance e Slow Start sono due algoritmi mutualmente esclusivi, cioè non possono mai avvenire contemporaneamente, tuttavia esiste un forte legame tra di loro. Senza scendere troppo nel tecnico, quando avviene una congestione e vengono persi segmenti durante lo Slow Start, viene salvata in una speciale variabile "ssthresh" (che sta per "slow start threshold") la metà del valore della congestion window per cui abbiamo perso segmenti. La finestra viene poi riportata ad uno, dopodichè, quando il valore della congestion window ricresce diventando uguale a ssthresh, la crescita della finestra diventa lineare e non più esponenziale. Quindi, se durante una slow start abbiamo una congestione per una congestion window pari a 32 byte, salviamo in ssthresh il valore 16, rimettiamo la congestion buffer a 1, poi rifacciamo una nuova slow start fino a 16 (crescita esponenziale: 1, 2, 4, 8, 16) dopodichè l'algoritmo di Congestion Avoidance interviene sulla velocità della crescita della congestion window (crescita lineare: 17, 18, 19, 20 ...).

**Fast Retrasmit** (ritrasmissione veloce): come dicevamo nel paragrafo delle finestre scorrevoli, quando un pacchetto viene perso, gli ACK dei segmenti successivi arrivati a destinazione riportano come Acknowledgement number il valore del segmento mancante nell'ordine di consegna. In condizioni normali dovremmo aspettare lo scadere del timer per il segmento andato perduto prima di poterlo ritrasmettere. Questo perchè forse il segmento incriminato non è veramente andato perduto, ma è semplicemente molto lento. L'algoritmo di Fast Retrasmit permette di scavalcare il timer del pacchetto sospetto quando sono già arrivati TRE ACK riportanti lo stesso valore nell'Acknowledgement number (sintomo che quel pacchetto, più che essere lento, ha probabilmente già preso il volo!).  
Facciamo un esempio per chiarire questo algoritmo.  
Supponiamo che il mittente spedisca i segmenti: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  
ma per un problema di trasmissione il segmento n.4 viene perso.  
Al ricevente arriveranno: 1, 2, 3, 5, 6, 7, 8, 9, 10  
Gli Ack. number spediti indietro al mittente saranno: 1, 2, 3, 3, 3, 3, 3, 3, 3  
La conclusione è che il segmento 4 è probabilmente andato perduto e che sono già arrivati altri 6 pacchetti dopo che è arrivato il 3. Con il Fast Retrasmit invece di ricevere un sacco di ACK tutti uguali aspettando che il timer di 4 scada, ritrasmettiamo il segmento 4 (valoreACKduplicati+1) al terzo ACK uguale.

**Fast Recovery** (recupero veloce): per migliorare l'efficienza del Fast Retrasmit, quando ritrasmettiamo un segmento che probabilmente è andato perduto, Fast Recovery fa in modo che il valore della congestion window non venga ridotto proprio del tutto, e che non venga effettuata una Slow Start, bensì una Congestion Avoidance. Il valore della congestion window infatti non viene riportato a 1, ma a sshtresh + 3, dove ssthresh è ancora settato alla metà della congestion window per cui è avvenuta la congestione, e 3 è il numero di ACK duplicati, questo per aumentare la c.window del numero di segmenti che sono nel buffer di ricezione del destinatario, inoltre per ogni ulteriore ACK uguale incrementa ancora la congestion window di uno, questo per aumentare la c.window del numero di segmenti partiti. Se permesso dal nuovo valore della c.window, viene trasmesso un nuovo pacchetto. Quando arriva il successivo ACK relativo a nuovi dati, setta la c.window a ssthresh (la metà). Ed eccoci in Congestion Avoidance.

IL PROTOCOLLO UDP

Esiste anche un fratello minore del TCP, e si chiama UDP, User Datagram Protocol.

E' usato in tutti quei casi in cui non è necessaria tutta la potenza di TCP, ad esempio quando dobbiamo inviare una richiesta che sta tutta in un solo datagramma. Infatti ogni datagramma UDP viene mandato in un singolo datagramma IP.

In sostanza l'unica cosa che un datagramma UDP aggiunge al protocollo IP è la capacità di usare le porte.

L'esempio classico è quando bisogna contattare un server DNS. Un server DNS, Domain Name Service, svolge un ruolo molto importante, specialmente in Internet: traduce indirizzi umani (www.xyz.com) in indirizzi IP (194.37.205.123). Una richiesta del genere è molto corta, non abbiamo bisogno di usare il TCP.

L'UDP non si divide in datagrammi e non tiene conto di ciò che viene mandato o di ciò che non è arrivato. Dopo tutto, se qualcosa va storto e non si riceve risposta, basta reinviare lo stesso pacchetto UDP. In gergo tecnico, si dice che TCP fornisce un servizio connection-oriented, mentre UDP è un protocollo connectionless (senza connessione).

L'header UDP è composto dai campi: porta sorgente/porta destinataria, equivalenti alle porte dell'header TCP, un campo Message Lenght che indica la lunghezza del datagramma incluso l'header, un campo di checksum di 16 bit ed infine il campo dati.

IL PROTOCOLLO ICMP

Questo protocollo è abbastanza particolare perchè si trova sullo stesso layer di IP (diversamente da TCP e UDP), ma usa comunque IP per spedire i suoi dati. ICMP non aggiunge garanzie a IP, quindi è possibile che i datagrammi ICMP non arrivino a destinazione o che arrivino corrotti (esattamente come per IP).

Il compito di ICMP è quello di inviare messaggi di servizio e messaggi di errore.

Un datagramma ICMP è composto dai campi: type (8 bit), che indica il tipo di messaggio, code (8 bit), un codice di errore che dipende dal tipo di messaggio, checksum (16 bit) per sapere se ci sono errori nel datagramma, data (variabile), che contiene informazioni sul messaggio e di solito include parte del datagramma IP che ha generato l'errore.

I principali codici sono:

0 - Echo Reply: in risposta all'ICMP n.8

3 - Destination Unreachable: viene generato da un router intermedio quando questo non è in grado di trovare il computer desiderato, oppure viene generato dal computer destinatario se si richiede un servizio (protocollo o porta) che non è attivo.

4 - Source quench: nelle congestioni, viene generato da un router intermedio quando questo ha esaurito lo spazio dove accodare i messaggi in arrivo, oppure viene generato dal computer destinatario se i datagrammi stanno arrivando troppo velocemente.

5 - Redirect: se viene generato da un router intermedio conterrà un indirizzo IP relativo ad un router che andrebbe contattato al posto di questo.

8 - Echo: usato per controllare se un computer è attivo. E' il messaggio che genera il comando ping.

9 - Router Advertisement: Con questo messaggio ICMP, i router periodicamente segnalano la loro presenza sulle reti a cui sono collegati.

10 - Router solicitation: Usato per forzare i router in ascolto ad identificarsi per non dover attendere i Router Advertisement periodici.

11 - Time exceeded: viene generato da un router intermedio quando il campo Time To Live del datagramma è zero. Se generato dal computer destinatario indica che è scaduto il timer che controlla il riassemblamento di un datagramma, il datagramma sarà scartato.

12 - Parameter problem: indica che è stato incontrato un problema durante l'analisi dei paramentri dell'header IP.

13 - Time Stamp request: usato per misurare le prestazioni e per il debugging ma mai per sincronizzare orologi. Il mittente salva in un campo dell'ICMP n.13 il suo time stamp. Il ricevente prepara un ICMP 14 settando altri due campi: il time stamp di ricezione e di trasmissione.

14 - Time Stamp reply: in risposta all'ICMP n.13

17 - Address mask request: Una richiesta broadcast che serve per scoprire la maschera subnet usata dalle macchine in rete. Usato nelle richieste Reverse-ARP, un particolare tipo di ARP usato dalle macchine che non possiedono dischi e non possono memorizzare i dati della rete tra uno spegnimento e la riaccensione. Tutti i riceventi del broadcast ICMP 17 rispondono preparando un ICMP 18 con i dati della sottorete.

18 - Address mask reply: in risposta all'ICMP n.17

IL PROTOCOLLO IP

IP è un protocollo un pò particolare: non garantisce che i suoi pacchetti, detti datagrammi IP, arrivino a destinazione, che siano privi di errori, o che essi arrivino nell'ordine corretto. Questo lavoro viene lasciato ai protocolli superiori come TCP. Per questo motivo IP è un protocollo abbastanza semplice.

L'header di IP si aggiunge a sua volta all'header di TCP visto in precedenza.

L'header IP

Contiene i seguenti campi; quelli in rosso sono i più importanti

* Version; (4 bit) la versione di IP usata. Vale 4 per l'IP standard e 6 per l'IPv6.
* IHL; (4 bit) IP Header Lenght; indica la lunghezza dell'header in parole di 32-bit.
* Type of Service; (8 bit) serve ad assegnare una priorità di consegna ai dati, ma è scarsamente adottato.
* Total Length; (16 bit) lunghezza totale del datagramma, header e dati, espressa in byte.
* Identification, Flags, Fragment Offset; (16+3+13 bit) sono usati per gestire e riunire i frammenti di datagramma. Spiego cos'è un frammento nel prossimo paragrafo.
  + Identification; I frammenti di un datagramma hanno lo stesso numero di Identification.
  + Flags; contiene i flag, 0, un flag riservato, DF (don't fragment), se attivo indica che il datagramma non deve essere frammentato, MF (more fragments), se è zero indica che è l'ultimo frammento.
  + Fragment Offset; contiene, misurato in unità di 8 byte, quanti dati sono presenti nei frammenti precedenti a questo. Se il frammento è il primo o è l'unico, il valore è zero.
* Time To Live; (8 bit) un numero che decresce ogni volta che viene attraversato un nuovo sistema. Se è zero il datagramma viene scartato. E' utile per evitare che un datagramma finisca in un loop infinito.
* Protocol Number; (8 bit) indica quale protocollo di alto livello riceverà il datagramma. Ad esempio per ICMP è 1, per IP è 4 (nei casi di incapsulamento), per TCP è 6, per UDP è 17.
* Header Checksum; (16 bit) per sapere se l'header contiene errori. E' indipendente dal checksum del TCP.
* Source Address/Destination Address; (32 bit / 32 bit) un campo molto importante, indica l'indirizzo IP del nodo sorgente/destinatario. Da non confondere con la source port e la destination port del livello di TCP.
* Option; (lunghezza variabile) alcune opzioni per l'IP;
* Padding; (lunghezza variabile) una serie di 0 inserita perchè la lunghezza dell'header IP sia un multiplo di 32 bit.

E alla fine

* Data; attenzione, qui non c'è solo il datagramma iniziale. Ricordate che prima abbiamo aggiunto anche l'header di TCP.

La frammentazione dei datagrammi IP

Un datagramma IP, quando viene passato al livello fisico, viene incapsulato in un header Ethernet per formare un frame fisico. Il problema è che le dimensioni massime di questo frame sono limitate. Il valore di questo limite è chiamato MTU, maximum transfer unit (unità massima di trasferimento). Cosa succede se il nostro datagramma IP è troppo grande per stare in una trama lunga al massimo MTU byte? La risposta a questo problema è la tecnica della frammentazione. Quello che succede è che il datagramma IP viene spezzettato in tanti frammenti abbastanza piccoli da entrare nel frame della rete. Ogni frammento possiede un proprio header molto simile all'header IP del datagramma originale, ma con i campi Identification, Flags, Fragment Offset settati opportunamente (per le loro funzioni vedere il paragrafo precedente). Innanzi tutto il flag DF non deve essere settato, altrimenti il datagramma non viene frammentato ma semplicemente scartato. Il flag MF è pari a uno in tutti i frammenti tranne l'ultimo. Il campo offset viene riempito con il valore opportuno per ogni frammento. Inoltre viene ricalcolato il contenuto di tutti quei campi dell'header IP contenenti valori variabili in funzione della lunghezza (checksum compreso).

Ora che i frammenti sono diventati nuovi datagrammi IP vengono inoltrati normalmente. Sul computer ricevente viene riassemblato il datagramma IP originale facendo uso di questi campi. La frammentazione è piuttosto fragile: la perdita di un solo frammento comporta la perdita dell'intero datagramma.

In generale avere un frame molto grande permette di avere una maggiore efficienza di trasmissione (anche se nei collegamenti soggetti a problemi di lentezza e corruzione dei pacchetti, avere un MTU grande sarà solo fonte di guai, dato che ogni pacchetto perso contiene molti dati). Normalmente si cerca di evitare di dover ricorrere alla frammentazione. La lunghezza finale del datagramma dipende sia dallo spazio occupato dagli header TCP e IP, che dalla lunghezza del segmento creato da TCP. Per evitare la frammentazione bisogna tenere conto di questi valori confrontandoli con il valore di MTU minimo tra quello delle due macchine. Anche così però non possiamo avere la garanzia che non avverrà una frammentazione, perchè è sempre possibile che alcune reti intermedie abbiano limiti MTU inferiori. E in alcuni casi avverrà anche più di una frammentazione.

Finito? Quasi... , ci siamo dimenticati un piccolo (?) particolare; stiamo usando una rete Ethernet, quindi dovremo aggiungere anche l'header di Ethernet. Lo trovate descritto per bene nella pagina [Analisi delle trame Ethernet](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/analisi.html).

Se non avete voglia di leggerlo vi dico brevemente che nell'header vengono aggiunti l'indirizzo Ethernet del mittente e del destinatario; fate attenzione: in Ethernettiano questo non è un indirizzo IP a 32 bit, ma quello della scheda di rete a 48 bit. Prima di proseguire è bene vedere cosa sono questi due indirizzi.

Indirizzi IP - Indirizzi Ethernet

Un indirizzo IP, 32 bit, viene indicato come 4 numeri decimali, ognuno esprime 8 bit (8bit x 4n=32bit), es. 192.168.150.10. Essendo solo 8 i bit per ogni numero, i valori andranno da 0 a 255.

Un indirizzo IP a 32 bit può essere visto come una coppia di due numeri: il numero di rete e il numero di host o nodo. Il numero di bit usato per il numero di rete dipende dalla classe di indirizzo. Esistono cinque classi di indirizzi IP:  
Classe A: inizia con un bit a 0 (primo ottetto da 1 a 126 in decimale), 7 bit per la rete , 24 per l'host. Permette di avere 126 reti con 16777213 host ciascuno.  
Classe B: inizia con due bit a 10 (primo ottetto da 128 a 191 in decimale), 14 bit per la rete , 16 per l'host. 16382 reti, di 65534 host ciascuno.  
Classe C: inizia con tre bit a 110 (primo ottetto da 192 a 223 in decimale), 21 bit per la rete , 8 per l'host. 2097150 reti, di 254 host ciascuno.  
Classe D: inizia con quattro bit a 1110 (primo ottetto da 224 a 239 in decimale), riservato per il multicasting  
Classe E: inizia con quattro bit a 1111 (primo ottetto da 240 a 254 in decimale), riservato per usi futuri

Il numero di rete è assegnato da un ente centrale, l'InterNIC, il numero di host è invece deciso dal possessore di quel numero di rete. Quando il numero di host è fatto solo da '0', l'indirizzo esprime l'indirizzo di rete. Quando è fatto di soli '1', indica un broadcast a tutti i nodi della rete.

Dato che la suddivisione per classi è piuttosto grezza, è stato creato il concetto di subnet, o sottorete, che permette di sottrarre qualche bit del numero dell'host in favore di una maggiore flessibilità di configurazione (ad esempio per separare il traffico in rete tramite un router), invisibile fuori dalla rete. In questo modo l'indirizzo è composto da: un numero di rete, un numero di subnet, un numero di host.

Se facciamo un AND bit a bit tra un indirizzo IP e una subnet mask quello che otteniamo è il numero di rete comprensivo del numero di subnet. I valori di default per le prime tre classi sono:  
Classe A: 255.0.0.0, pari a 11111111.00000000.00000000.00000000  
Classe B: 255.255.0.0, pari a 11111111.11111111.00000000.00000000  
Classe C: 255.255.255.0, pari a 11111111.11111111.11111111.00000000

Usando queste maschere standard con un indirizzo IP, riotteniamo semplicemente il numero di rete. Se però volessimo fare tre sottoreti, dato un numero di rete di Classe C, possiamo "rubare" tre bit al quarto ottetto (l'inizio del numero di host), bastano 3 bit perchè con 111 abbiamo il numero 7(>5). Quando facciamo questo otteniamo una Subnet mask di  
11111111.11111111.11111111.**111**00000 (255.255.255.224)  
Se la rete è 193.1.1.0, gli host delle varie sottoreti avranno indirizzi che iniziano per:  
11000001.00000001.00000001.00000001.**000**00000 (193.1.1.0) 1° subnet  
11000001.00000001.00000001.00000001.**001**00000 (193.1.1.32) 2° subnet  
11000001.00000001.00000001.00000001.**010**00000 (193.1.1.64) 3° subnet  
11000001.00000001.00000001.00000001.**011**00000 (193.1.1.96) 4° subnet  
11000001.00000001.00000001.00000001.**100**00000 (193.1.1.128) 5° subnet  
11000001.00000001.00000001.00000001.**101**00000 (193.1.1.160) 6° subnet  
11000001.00000001.00000001.00000001.**110**00000 (193.1.1.192) 7° subnet  
11000001.00000001.00000001.00000001.**111**00000 (193.1.1.224) 8° subnet

In realtà le subnet 1=000 e 8=111 è meglio non usarle perchè i numeri fatti di soli 0 o soli 1 hanno le funzioni particolari di cui parlavamo prima.

N.B. : quando assegnate gli indirizzi IP alle vostre macchine, è buona norma non assegnare mai valori già usati in Internet, per questo sono stati definiti indirizzi "sicuri": 10.0.0.0 (numero di rete=primi 8 bit), 172.16.0.0 (numero di rete=primi 20 bit), 192.168.0.0 (numero di rete=primi 16 bit), se non avete particolari esigenze io vi consiglio quest'ultimo. Evitate come al solito i valori 0 e 255. Un'altra classe di indirizzi riservata è 127.0.0.0 (numero di rete=primi 8 bit), che identifica il localhost, ossia il proprio computer. In generale il mio consiglio è di assegnare alle macchine della vostra LAN gli indirizzi 192.168.x.y dove x rappresenta il numero di sottorete, e y il numero per il nodo, e come subnet 255.255.255.0. Tra parentesi, una macchina può essere collegata a più di una rete, avrà quindi due o più indirizzi IP, uno per ogni rete. Tale macchina viene detta multi-homed e può svolgere le funzione di un router (un altro nome usato è gateway, tuttavia questa parola ha anche altre definizioni e può generare confusione).

Un indirizzo Ethernet, 48 bit, viene invece indicato con una notazione differente: 6 numeri esadecimali, ma ognuno di essi continua ad esprime ancora 8 bit(8bit x 6n=48bit), es. 20-53-52-b8-1f-00. I valori vanno da 00 a ff (che per l'appunto sono poi equivalenti in decimale a 0 e 255, cambia solo il modo di scriverli). Anche qui i numeri 0 e ff non vanno usati, ma di questo non vi dovete preoccupare. Infatti gli indirizzi Ethernet sono già scritti nelle schede di rete quando vengono fabbricate. Essi devono seguire le disposizioni della IEEE in materia, secondo le quali non devono esistere due schede di rete con lo stesso indirizzo.

Ora la domanda sorge spontanea: cosa unisce l'indirizzo IP di una macchina con l'indirizzo fisico della scheda di rete di quella macchina?

Nulla! E' per questo che hanno inventato un altro protocollo molto importante: ARP, Address Resolution Protocol.

IL PROTOCOLLO ARP

Quando due computer su una stessa rete Ethernet vogliono comunicare, essi devono prima conoscere l'indirizzo fisico. Ogni volta che usiamo TCP/IP su Ethernet e vogliamo comunicare con un sistema di cui conosciamo solo l'indirizzo IP, viene spedita una richiesta ARP di tipo broadcast sulla rete. In essa si chiede ai computer in ascolto quale sia l'indirizzo fisico corrispondente al quell'indirizzo IP. Il computer interessato fornisce la risposta, noi la riceviamo, la mettiamo nella nostra ARP-Table, dopo di che possiamo parlare con quel computer in modo diretto. Se in seguito abbiamo nuovamente bisogno di parlargli, guardando nella ARP-Table, ci accorgeremo che conosciamo già l'indirizzo fisico, così non dovremo neppure inviare una richiesta ARP.

Già che siamo in argomento, volevo farvi notare una curiosità che forse non tutti conoscono (ed al tempo stesso vedrete ARP al lavoro sul vostro Pc). Quando siete in Internet, cioè adesso, lanciate da una finestra Dos il comando "arp -a"; questo comando serve per vedere la ARP-Table. Scoprirete una cosa strana: tutti gli indirizzi IP dei siti che avete visitato di recente, hanno l'indirizzo fisico uguale! Perchè? Perchè in realtà quell'indirizzo fisico corrisponde al router del vostro provider. Questo perchè quando cercate di accedere ad un sito dovete lasciare tutto in mano al router, il quale fare tutto il lavoro per voi, come se fosse un maggiordomo virtuale.

Ora che abbiamo tutto pronto, e che conosciamo grazie ad ARP l'indirizzo fisico del destinatario, non resta che spedire il tutto!

Nel sistema ricevente, tutti gli header saranno analizzati ed utilizzati opportunamente e via via rimossi.

I protocolli

NetBEUI, NETBIOS e SMB

[Introduzione](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "introduzione)

[SMB](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "smb)

[CIFS](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "cifs)

[NetBIOS e NetBEUI](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "netbios_netbeui)

[I limiti](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "limiti)

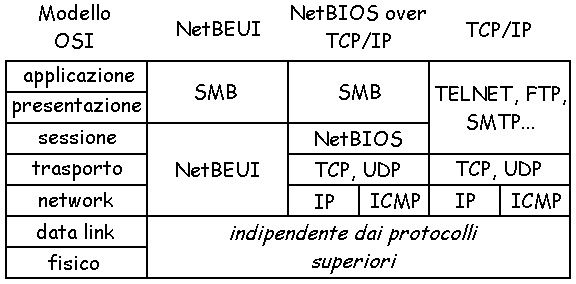
[NetBIOS over TCP/IP](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "nbt)

[Cenni sulla sicurezza](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "sicurezza)

[Esempio di comunicazione NBT via Internet](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/netbeui.html" \l "esempio)

INTRODUZIONE

Come per il TCP/IP iniziamo subito con un pò di chiarimenti per quel che riguarda i nomi e i ruoli di questa famiglia di protocolli. NetBEUI è l'acronimo di "NetBios Extended User Interface", cioè, interfaccia utente estesa di NetBIOS. NetBIOS a sua volta significa "Network Basic Input Output System", ossia, sistema input/output di base di rete. Infine SMB sta per "Server Message Block", blocco messaggio server. Ora che questi acronimi e le relative traduzioni vi hanno sufficientemente confuso le idee (o lasciato indifferenti), guardiamo in modo più chiaro il ruolo di ognuno di questi protocolli aiutandoci con la figura seguente. Se non sapete cosa sono i layer vi consiglio di vedere la pagina [Il viaggio dei dati](https://www.deltabeta.it/2010-2016/corso_rete/rete_ueg/viaggio.html). Con l'aiuto della figura chiariamo subito la confusione tra NetBIOS e NetBEUI.



Più che un protocollo, il NetBIOS, che lavora al livello Sessione, è un'API, cioè un'interfaccia di programmazione che, attraverso un set di comandi standard, unisce l'SMB con i protocolli di trasporto ed instradamento sottostanti, come TCP/IP o IPX/SPX.

NetBEUI è invece un protocollo, ed ingloba in modo nativo sia l'interfaccia NetBIOS, sia una semplice funzionalità di trasporto, il livello di trasporto del NetBEUI implementa il protocollo OSI LLC2. Quindi, mentre il NetBIOS può lavorare solo se è abbinato ad un protocollo di trasporto, NetBEUI non ha bisogno ne di protocolli di trasporto ne di interfacce verso SMB.

SMB

Partiamo dall'SMB. Creato dall'IBM a metà dagli anni 80 e successivamente adottato e modificato dalla Microsoft, si tratta di un importante protocollo, la cui implementazione è presente in quasi tutti i sistemi Windows.

Come si vede dalla figura precedente, si tratta del protocollo di più alto livello, al di sotto del quale si può trovare il NetBEUI oppure il NetBIOS (quest'ultimo a sua volta su un protocollo di trasporto quale il TCP/IP o l'IPX/SPX).

Il suo funzionamento è client-server, del tipo richiesta-risposta, dove il server è quel sistema che rende disponibili le proprie risorse condivise al client. I messaggi di richiesta e di risposta tra il client e il server sono detti (guardacaso) SMB.

Essenzialmente il lavoro di questo protocollo è quello di rendere possibile la condivisione di file e stampanti, incluse tutte le operazioni che comunemente vengono fatte su queste risorse (per es: aprire, chiudere, leggere, scrivere, creare, cancellare...). Come dicevamo, ad ognuna di queste operazioni corrisponde un certo messaggio SMB (open, read, write, close...). SMB è anche il responsabile del tanto comodo quanto problematico "browsing" delle risorse di rete.

Fa anche parte del protocollo SMB quel particolare elemento che permette di disporre delle risorse remote come se fossero locali. Il nome di questo componente è "redirector". Tramite il redirector ad esempio, è possibile vedere un disco di un altro computer come se fosse un disco sul proprio computer.

Per adattare meglio l'SMB ai vari ambienti sotto cui può lavorare, sono state create diverse varianti di questo protocollo. Per questo, quando due computer iniziano una connessione SMB, la prima cosa che viene fatta è decidere quale variante usare. Il modo per mettersi d'accordo è di inviare come primo messaggio, un SMB particolare chiamato "negprot" (negozia protocollo), che contiene le varianti conosciute dal mittente. Il ricevente risponderà indicando una certa variante oppure, se non ne conosce nessuna tra quelle elencate dal mittente, risponderà con un numero speciale che indica l'errore. Successivamente ha luogo l'autenticazione, che avviene spedendo al server un nome utente e una password. In questo caso l'SMB usato è "sesssetupX". Se il login ha successo, viene inviato come risposta al richiedente un numero, l'UID, che dovrà essere reinviato al server in tutte le successive connessioni con esso. Infine, per accedere ad una risorsa condivisa, si fa uso dell'SMB "tconX" a cui segue in risposta un altro numero, il TID da usare in modo analogo all'UID nei successivi accessi a quella risorsa.

Il protocollo SMB è in grado di gestire due livelli di sicurezza: share e user. In modalità share viene associata una password ad ogni risorsa condivisa. Viene permesso l'accesso anche a tutti i file presenti all'interno della risorsa. In modalità user, il client deve essere innanzi tutto autenticato dal server tramite una coppia utente-password indipendenti dalle risorse che l'utente vorrà usare. L'accesso alla risorsa è deciso in base ai privilegi che il client possiede. Si definisce dominio, un insieme di computer, in cui la gestione della sicurezza è affidata ad un'entità centrale, chiamata PDC, primary domain controller, controllore primario del dominio, al cui fianco possono esserci controller secondari di backup. Il PDC contiene ad esempio i nomi dei client, le relative password, il gruppo di appartenenza ecc. .

Le comunicazioni con il livello inferiore (NetBEUI o NetBIOS) avvengono tramite particolari strutture dati di 64 byte, chiamate NCB, cioè "Network Control Block", blocco di controllo di rete. Attraverso le NCB passano le richieste e le risposte SMB. Il metodo consiste nel caricare opportunamente queste strutture dati con tutte le informazioni necessarie (codice del comando, nomi NetBIOS, puntatori e parametri vari). L'accesso alle NCB può essere asincrono, se il controllo ritorna subito al programma, o sincrono, se invece il programma attende il completamento dell'operazione specificata.

CIFS

CIFS, o Common Internet File System, è una novità su cui sta lavorando Microsoft insieme ad altre società. Si tratta di un protocollo basato su SMB, ma orientato verso Internet e mira ad aggiungersi ai sistemi classici di accesso ai file (FTP, HTTP, NFS). Le specifiche sono aperte: l'intento è che diventino un RFC e vengano implementate su tutte le piattaforme, incluse quelle Microsoft e Unix. I vantaggi dichiarati consisterebbero in un accesso multiplo in scrittura ai file senza perdita di integrità, robustezza, buona velocità, sicurezza negli accessi, supporto dei caratteri in formato Unicode, uso di nomi con significatività globale per i file.

NETBIOS e NETBEUI

Il NetBIOS ed il NetBEUI sono stati sviluppati rispettivamente nel 1983, dalla Sytek per IBM, e nel 1985 da IBM. Ancora oggi restano molto diffusi nei sistemi Microsoft, in virtù dei loro pregi nell'ambito delle reti locali.

Per essere più precisi, l'implementazione del NetBEUI nei sistemi Microsoft differisce dalle ultime versioni del NetBEUI standard (NetBEUI 3.0) per alcuni aspetti, pur preservandone la compatibilità. Ad esempio, il NetBEUI di Microsoft, chiamato NBF (NetBIOS Frame), presenta alcune migliorie rispetto al protocollo standard, ed invece di interfacciarsi con i livelli superiori tramite il consueto NetBIOS, utilizza un'altra interfaccia più flessibile, chiamata TDI, Transport Driver Interface, cioè interfaccia del dispositivo di trasporto.

All'epoca, l'obiettivo era infatti quello di creare un protocollo su misura per le LAN di dimensioni contenute (fino a circa 200 nodi), quindi doveva essere piccolo, semplice, veloce e doveva permettere di assegnare nomi "umani" alle risorse, invece dei "complessi" indirizzi usati dal TCP/IP. Inoltre NetBIOS è stato progettato perchè usasse intensamente i broadcast (messaggi uno a tutti), piuttosto che interrogare un'entità centralizzata. In questa logica, un computer che in seguito all'accensione voglia registrarsi con il proprio nome NetBIOS come nuovo nodo, deve inviare appositi messaggi broadcast per farsi accettare come nuovo nodo attivo. Se il nome è già usato, il precedente possessore di quel nome deve inviare un messaggio di risposta per respingere la scelta. Se invece un computer vuole localizzare un nodo preesistente, invierà ancora altri broadcast nel tentativo di contattare quello specifico nodo. Il computer remoto è sempre tenuto a rispondere ai broadcast con un messaggio unicast (uno a uno). Solo successivamente i due computer potranno comunicare senza usare broadcast.

I nomi usati da NetBIOS sono fatti di 16 caratteri alfanumerici e possono essere di due tipi: ci sono i nomi unici, che come dice il nome, possono essere usati per identificare al più una sola risorsa in rete, ed i nomi di gruppo che invece si associano a più nomi unici. In ambiente Windows, un nome unico può individuare sia un computer della rete, sia anche un servizio offerto da un computer (una certa applicazione o una funzionalità particolare), ma dato che NetBIOS non dispone dei numeri di porte, come quelle del TCP/IP, per poter identificare queste funzionalità viene riservato il 16esimo carattere, che diventa un suffisso e quindi limita il nome a soli 15 caratteri. Ecco una lista dei suffissi usati da Microsoft.

Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Number\_\_\_\_Type\_\_\_\_\_\_\_Service  
==========================================================================  
<computername>\_\_\_\_\_\_\_\_00\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Workstation Service  
<computername>\_\_\_\_\_\_\_\_01\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Messenger Service  
<\_\_MSBROWSE\_\_> \_\_\_\_01\_\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Master Browser  
<computername>\_\_\_\_\_\_\_\_03\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Messenger Service  
<computername>\_\_\_\_\_\_\_\_06\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_RAS Server Service  
<computername>\_\_\_\_\_\_\_\_1F\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_NetDDE Service  
<computername>\_\_\_\_\_\_\_\_20\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_File Server Service  
<computername>\_\_\_\_\_\_\_\_21\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_RAS Client Service  
<computername>\_\_\_\_\_\_\_\_22\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Exchange Interchange   
<computername>\_\_\_\_\_\_\_\_23\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Exchange Store  
<computername>\_\_\_\_\_\_\_\_24\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Exchange Directory  
<computername>\_\_\_\_\_\_\_\_30\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Modem Sharing Server Service  
<computername>\_\_\_\_\_\_\_\_31\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Modem Sharing Client Service  
<computername>\_\_\_\_\_\_\_\_43\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_SMS Client Remote Control  
<computername>\_\_\_\_\_\_\_\_44\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_SMS Admin Remote Control Tool  
<computername>\_\_\_\_\_\_\_\_45\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_SMS Client Remote Chat  
<computername>\_\_\_\_\_\_\_\_46\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_SMS Client Remote Transfer  
<computername>\_\_\_\_\_\_\_\_4C\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_DEC Pathworks TCPIP Service  
<computername>\_\_\_\_\_\_\_\_52\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_DEC Pathworks TCPIP Service  
<computername>\_\_\_\_\_\_\_\_87\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Exchange MTA  
<computername>\_\_\_\_\_\_\_\_6A\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Exchange IMC  
<computername>\_\_\_\_\_\_\_\_BE\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Network Monitor Agent  
<computername>\_\_\_\_\_\_\_\_BF\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Network Monitor Apps  
<username> \_\_\_\_\_\_\_\_\_\_\_03\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Messenger Service  
<domain> \_\_\_\_\_\_\_\_\_\_\_\_\_00\_\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Domain Name  
<domain> \_\_\_\_\_\_\_\_\_\_\_\_\_1B\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Domain Master Browser  
<domain> \_\_\_\_\_\_\_\_\_\_\_\_\_1C\_\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Domain Controllers  
<domain> \_\_\_\_\_\_\_\_\_\_\_\_\_1D\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Master Browser  
<domain> \_\_\_\_\_\_\_\_\_\_\_\_\_1E\_\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Browser Service Elections  
<INet~Services> \_\_\_\_\_\_\_1C\_\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Internet Information Server  
<IS~Computer\_name>\_\_\_\_00\_\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Internet Information Server  
<computername>\_\_\_\_\_\_\_\_[2B]\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_Lotus Notes Server  
IRISMULTICAST \_\_\_\_\_[2F]\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Lotus Notes  
IRISNAMESERVER \_\_\_\_[33]\_\_\_\_\_\_\_G\_\_\_\_\_\_\_\_\_Lotus Notes  
Forte\_$ND800ZA \_\_\_\_\_[20]\_\_\_\_\_\_\_U\_\_\_\_\_\_\_\_\_DCA Irmalan Gateway Service

Numero = suffisso, in esadecimale  
U = nome unico  
G = nome gruppo  
Se volete vedere la tabella dei servizi NetBIOS presenti in un computer, lanciate da una shell Dos il comando  
nbtstat -a <nome del computer>  
oppure  
nbtstat -A <indirizzo IP del computer>

il perchè si possa usare un IP in ambiente NetBIOS sarà più chiaro dopo aver letto la sezione 'NetBIOS over TCP/IP'.

Per evitare che ogni nodo debba contenere una copia completa della lista dei nomi registrati e delle relative risorse condivise, viene eletto tra i computer un Browse Master, a cui verrà affidata una unica lista. In generale l'elezione del browse master viene vinta dai sistemi di classe "superiore", come sistemi NT o macchine Linux su cui gira Samba (un server smb). Se c'è un Primary Domain Controller, sarà questo il browse master del dominio controllato. Tra sistemi Win9x, si può decidere manualmente chi sarà il browse master andando nel computer prescelto e selezionando: Pannello di Controllo->Rete->Condivisione file e stampanti-> Browse Master->attivato.

Le comunicazioni sono di tre tipi: sessioni, datagrammi, broadcast. Le sessioni si usano per scambiare con un altro nodo rilevanti quantità di dati con rilevamento e correzione d'errore. I datagrammi per inviare ad uno o più nodi (quando il destinatario è un gruppo) messaggi di dimensioni modeste ma senza dover stabilire una sessione. I broadcast infine, si usano per comunicare messaggi di dimensioni modeste a tutti i nodi in ascolto. In sessione è possibile inviare fino a 64Kb di dati per messaggio. Con i datagrammi si è limitati a 512byte per messaggio ed inoltre non si ha la garanzia che i dati arrivino. Tuttavia, negli ambiti in cui queste limitazioni non costituiscono un problema insormontabile, lavorare con i datagrammi rimane molto semplice e veloce.

Per separare il traffico in una stessa rete è stata sviluppata un'apposita funzionalità, chiamata "NetBIOS Scope ID". Si tratta di una stringa di caratteri aggiunta alla fine del nome. Due computer devono avere lo stesso scope ID per comunicare. Usando questa funzione è possibile ad esempio assegnare lo stesso nome a due nodi, se questi possiedono uno scope ID differente. In Windows9x per configurare lo scope ID, se avete un server WINS, bisogna andare in Pannello di Controllo->Rete->Tcp/Ip->Proprietà->Wins->area di validità ID. In mancanza di un server WINS si può modificare il registro di Windows con il programma regedit.exe. Fate attenzione a non usare regedit con troppa leggerezza, perchè se non si sta attenti è facile creare seri problemi. La chiave da trovare è: HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\VXD\MSTCP. Lo scope ID si aggiunge cliccando su modifica, poi nuovo, stringa e scrivere "ScopeID" (senza apici). Poi fare doppio click su questo nome e aggiungere come valore lo ScopeID prescelto.

I limiti

La diffusione di Internet ha messo subito in luce quali sono i limiti di NetBIOS e del suo socio NetBEUI quando la rete cresce di dimensioni.

Tralasciando problemi seri ma non insormontabili come il degrado prestazionale sulle WAN legato all'uso dei broadcast, rimangono tuttavia altri problemi di non facile soluzione: innanzitutto c'è il problema dell'unicità nomi. Dato che due computer non possono usare due nomi uguali nella stessa rete, bisognerebbe trovare un nome diverso per ogni computer connesso, cosa non banale in una rete geografica. In secondo luogo, sempre a causa dei nomi, NetBEUI non permette il routing. Cioè, dato un nome NetBIOS, è impossibile sapere quale sia la strada per raggiungerlo. Così, non possiamo sapere se il computer a cui vogliamo collegarci sia nel palazzo accanto o in un altro continente. Il problema in questo caso risiede nel fatto che i nomi NetBIOS non contengono alcuna informazione gerarchica. Per portare un esempio di nomi gerarchici basta prendere il TCP/IP, i cui nomi sono gerarchici da sinistra a destra: indirizzi del tipo xxx.yyy.zzz.www, si possono pensare come indirizzi postali del tipo: stato.città.strada.casa. Quando la rete diventa Internet, è immediato rendersi conto che un indirizzo IP sia molto più comodo e molto più versatile di un nome normale (nb: i nomi dei siti Internet corrispondono ad indirizzi IP e non sono nomi NetBIOS. Grazie ai server DNS infatti, collegandosi con [ftp.microsoft.com](http://www.xyx.it) è esattamente come collegarsi a 198.105.232.1). Un ultimo problema, forse il più importante, è che i router di Internet non permettono il propagarsi dei broadcast, tanto cari al NetBIOS quando cerca di localizzare un nodo.

Per dire tutto in tre parole: NetBIOS non scala.

Per evitare che il NetBIOS affondasse, invece di navigare in Internet, tornò molto comodo a Microsoft il fatto di poter abbinare l'interfaccia NetBIOS ad un protocollo molto più flessibile del NetBEUI, come il TCP/IP, e limitando l'impiego del NetBEUI nell'ambito delle reti locali tra sistemi Microsoft. In questo modo ogni messaggio elaborato da NetBIOS viene incapsulato in un messaggio TCP/IP, che non soffre delle limitazioni di cui parlavamo prima. Chiaramente i problemi non spariscono in modo indolore, anzi sorgono nuove complicazioni, e quello che la Microsoft ne tirò fuori fu una versione distorta di Internet, dove il DNS diventa WINS, e HOSTS diventa LMHOSTS. Vediamo un pò cosa è successo.

NetBIOS over TCP/IP

Per rendere possibile l'interfacciamento tra NetBIOS e TCP/IP è necessario che NetBIOS legga nomi, mentre TCP/IP deve leggere numeri. Ecco che nasce il "NetBIOS over TCP/IP", chiamato anche NBT, descritto negli RFC 1001 e 1002. Esso è in grado di lavorare su reti geografiche, basandosi su associazioni tra nomi NetBIOS e indirizzi IP fornitegli dall'esterno.

Iniziamo col dire che i nodi, in ambiente NBT, possono operare seguendo quattro modalità standard:

* b-nodi (nodi broadcast): usano broadcast sia per la registrazione che per la risoluzione dei nomi in indirizzi IP. I b-nodi soffrono ancora del problema dei broadcast di NetBIOS: se i nodi sono separati da router, non riusciranno a vedersi.
* p-nodi (nodi punto-punto): scoprono l'IP della risorse interrogando con richieste unicast un server WINS noto. WINS, Windows Internet Name Service, servizio nomi Internet Windows, è un server, che imitando i server DNS, fornisce una lista di coppie nome\_NetBIOS-indirizzo\_IP.
* m-nodi (nodi misti): usano prima il broadcast come gli m-nodi e poi le richieste come i p-nodi.
* h-nodi (nodi ibridi): l'opposto degli m-nodi, cioè prima contattano un server WINS noto, poi, se la richiesta ha esito negativo, fanno broadcast.

Una volta ricavati gli indirizzi IP al posto dei nomi, si possono aggirare i limiti descritti in precedenza ed è così possibile contattare nodi remoti.

I client Microsoft usano un sistema leggermente più complicato. Consultano in ordine: una cache interna, il server WINS se presente, poi fanno broadcast, e per finire leggono file LMHOSTS.

LMHOSTS è simile al file HOSTS delle macchine UNIX. E' un file da editare manualmente contenente, come una specie di server WINS statico, coppie di nomi\_NetBIOS-indirizzi\_IP. Torna utile se non non si dispone di nessun server WINS e non si riesce a risolvere un nome remoto tramite broadcast perchè il nodo è al di la di uno o più router. In realtà LMHOSTS può tornare utile anche se il destinatario risiede nella stessa rete locale. Infatti, per non essere obbligati a subire una marea di broadcast sulla propria rete, è possibile precaricare le associazioni del file LMHOSTS più usate nella cache, semplicemente mettendo alla fine di ogni linea un tag: "#PRE" (senza apici). Il file LMHOSTS si trova in genere in [C:\Windows](C:\\Windows). Se non l'avete mai usato avrà estensione .sam (cioè sample ossia esempio), per renderlo attivo dovete togliere il .sam. Nel file d'esempio, c'è anche una lista di tutti gli altri tag utilizzabili, con la relativa spiegazione.

Cenni sulla sicurezza

Nel protocollo TCP/IP, per stabilire una comunicazione, è necessario definire le porte a cui inviare le richieste. NBT usa perlopiù le seguenti porte:

137: risoluzione dei nomi NetBIOS (UDP)  
138, 139: datagrammi (UDP)  
139: sessioni (TCP)

Ricordo che per evitare che le proprie cartelle siano accessibili dall'esterno è bene togliere il binding tra TCP/IP di Accesso Remoto e Condivisione file e stampanti. Nella tabella di qualche pagina fa questo corrisponde al servizio con codice n.20 (File Server Service). Per restare in tema di sicurezza, tra le varie opzioni del TCP/IP di Accesso Remoto, c'è un'opzione chiamata "attiva NetBIOS su TCP/IP" che però nelle ultime versioni di Win95 e in Win98 non è deselezionabile. Questo non è comunque un grande rischio se avete tolto il binding di cui parlavo prima. L'unica cosa visibile dall'esterno sarà al massimo il nome del computer ed il vostro gruppo di lavoro, che in genere non rappresentano un grande problema di sicurezza (specialmente se il nome è qualcosa del tipo MIOPC1 e il gruppo è MIOGRUPPO!). Se invece avete la necessità di rendere accessibili le vostre cartelle a qualcun'altro su Internet, ricordatevi che non c'è nulla come una buona password! Per non attrarre i lamer di passaggio potete anche nasconderle mettendo il carattere "$" (senza apici) in fondo al nome della risorsa condivisa, tuttavia questa soluzione non è sicura se usata come unica difesa.

Se però avete serie esigenze di sicurezza e volete togliere le 3 porte NetBIOS 137, 138, 139, vi consiglio caldamente di tagliare la testa al toro installando un buon firewall.

Esempio di comunicazione NBT via Internet

Ricapitoliamo ciò che è stato detto finora portando un esempio di comunicazione NBT via Internet. Supponiamo di conoscere un computer che permetta l'accesso alle sue risorse condivise via Internet (cioè con la porta 139 aperta e con il TCP/IP di Internet in binding al servizio di condivisione). Per collegarci possiamo aggiungere al file LMHOSTS la linea  
xxx.yyy.zzz.www <...un pò di spazi...> NOMECOMPUTER <...un pò di spazi...> #PRE

dove xxx.yyy.zzz.www è l'IP di quel computer e NOMECOMPUTER è il suo nome NetBIOS (scopribile con nbtstat -A <indirizzo IP>). Nota: in qualche caso avrete bisogno di collegarvi, non tanto al nodo bensì ad un suo servizio. La linea da aggiungere è simile, ma il nome finisce con uno dei caratteri esadecimali visti in precedenza e specificabile con il tag \0x e tutto racchiuso tra virgolette, per esempio:  
xxx.yyy.zzz.www "NOMECOMPUTER \0x20" #PRE

per vedere le risorse condivise da quel nodo si può lanciare da una shell di Dos il comando  
net view [\\NOMECOMPUTER](https://NOMECOMPUTER)

e per installare una risorsa tra quelle elencate, come se fosse un drive del proprio computer  
net use x: [\\NOMECOMPUTER\RISORSA](https://NOMECOMPUTER/RISORSA)

dove "x:" è la lettera dell'unità che conterrà la risorsa (non mettete una lettera già in uso).