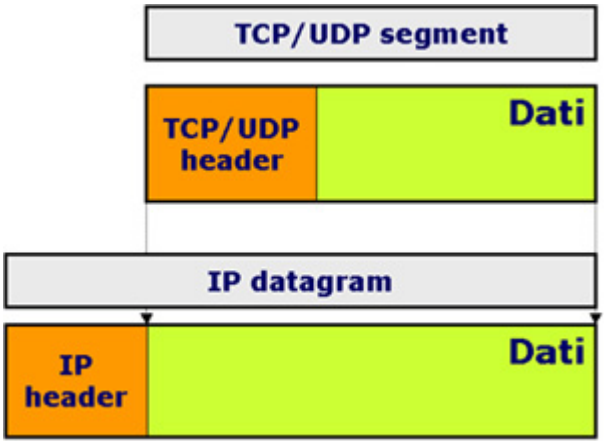
Protocolli di trasporto in Internet
TCP - Transmission Control Protocol



Il *Transmission Control Protocol* (**TCP**) è stato progettato al fine di offrire un servizio *end-to-end* perfettamente affidabile alle applicazioni, tenendo conto che la rete sottostante (IP) non è affidabile. Il TCP accetta dal livello superiore messaggi di lunghezza illimitata, li segmenta in pacchetti di piccole dimensioni e li invia in datagrammi.

Il protocollo è descritto nelle ***Request For Comments* (RFC) 793** ed è successivamente ampliato a causa di alcune *bug fixes*: RFC 1122. Se ne trova anche una estensione: RFC 1323.

Le funzioni svolte dal protocollo TCP sono:

- **controllo di errore**;
- **controllo di flusso**;
- **controllo di sequenza**;
- *multiplexing* delle connessioni su un singolo indirizzo di rete.

TCP riceve i dati a flussi dai protocolli di strato superiore che li inviano a *byte*, uno alla volta; quando arrivano allo strato TCP, i *byte* vengono raggruppati in **segmenti** TCP, che vengono quindi passati a IP per essere trasmessi alla destinazione successiva. La lunghezza dei segmenti è determinata da TCP.

Le funzionalità del protocollo TCP vengono garantite mediante la numerazione dei datagrammi e l'invio di messaggi di riscontro (**acknowledgment**) da parte della destinazione ogniqualvolta viene ricevuto correttamente il giusto datagramma della sequenza. Nel caso di connessioni interattive bidirezionali si usa la tecnica **piggybacking** (*acknowledgment* contenuto nelle risposte). Inoltre, i numeri di sequenza servono a TCP per il risequenziamento dei segmenti, qualora questi giungano alla destinazione finale in ordine errato. TCP adotta una tecnica di riconoscimento globale, che comprende tutti i *byte* fino al numero di riconoscimento meno uno.

Il modulo TCP ricevente può anche eseguire il controllo del flusso dei dati del mittente, molto utile per evitare la perdita di dati per superamento della capacità del **buffer** e l'eventuale saturazione della macchina ricevente. Il meccanismo si basa sull'emissione di un valore di finestra alla stazione trasmittente, la quale può inviare un numero specificato di *byte* all'interno di tale finestra; al raggiungimento di questo numero, la finestra viene chiusa e l'entità trasmittente deve interrompere l'invio dei dati.

Ogni trasmissione di dati deve essere preceduta da una fase di attivazione della connessione e seguita da una fase di rilascio.

TCP - Multiplazione

Compito di TCP è quindi anche quello di distinguere tra i diversi programmi applicativi e i diversi utenti che fanno uso di uno stesso sistema. Come avviene per UDP, si è stabilito che ogni sistema contenga un insieme di punti di destinazione chiamati **porte**. Anche in TCP, ogni porta è identificata da un intero positivo. L'indirizzo di un utente di strato TCP è denominato porta, mentre l'indirizzo completo nell'insieme dei protocolli TCP e IP è denominato **socket** ed è costituito dalla coppia:

- `porta@IP_Address`.

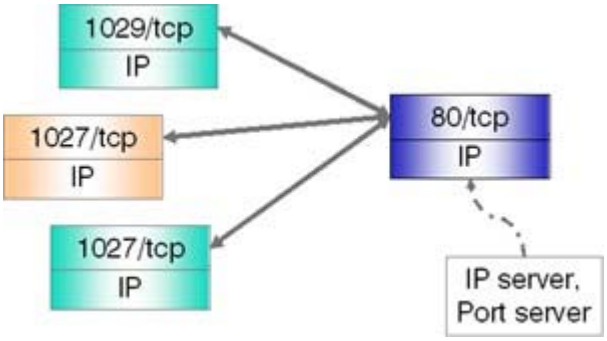
La componente *port* è contenuta nell'intestazione dell'unità di dati di TCP, mentre la componente *IP_Address* è contenuta nell'intestazione dell'unità dati di IP. Questo significa che tutte le sessioni di comunicazione in atto tra due specifici sistemi useranno lo stesso indirizzo IP di sorgente e lo stesso indirizzo IP di destinazione; saranno perciò distinte solo allo strato TCP.

Ne segue che queste sessioni sono multiplate su un unico indirizzo IP, ovvero su un unico canale IP di comunicazione (non su una connessione IP; in questo caso la definizione di multiplazione va usata con cautela dal momento che IP è un protocollo senza connessione).

TCP - Connessione

Come per **UDP** esistono dei numeri di porta ben noti (**well known port**), ma a differenza di UDP, alla stessa porta può corrispondere più di un processo. Tale maggiore complessità deriva dal fatto che TCP è un protocollo con connessione.

In TCP una connessione è identificata da una coppia di **socket**, relativa ai due processi che hanno stabilito la connessione.



Ad esempio una connessione tra la porta 2772 dell'*host* 197.2.7.2 e la porta 80 dell'*host* 151.80.4.1 sarà identificata dalla coppia:

- 2772@197.2.7.2, 80@151.77.4.1.

Grazie a tale meccanismo, un indirizzo di porta di un sistema può supportare connessioni multiple; la porta 2772 dell'*host* 197.2.7.2 potrebbe gestire contemporaneamente le seguenti connessioni (ed anche altre):

- 2772@197.2.7.2, 80@151.77.4.5;
- 2772@197.2.7.2, 80@165.20.2.3

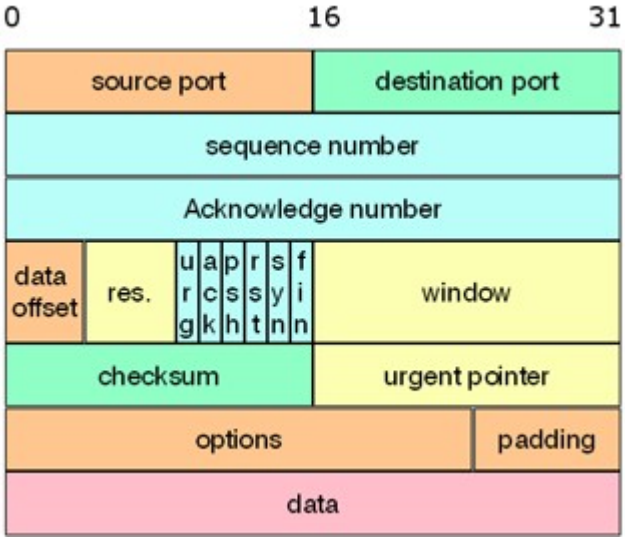
TCP - Conclusioni

Evidentemente tutte le funzionalità fornite da TCP hanno un costo in termini di aumento del ritardo

di trasmissione e di aumento della quantità di informazioni che devono essere trasferite nella rete (*overhead*); ciononostante TCP può essere impiegato anche in reti ad alta velocità. In alcuni esperimenti si sono raggiunte portate utili di 8 Mbit/s su una *Ethernet* con capacità di trasferimento pari a 10 Mbit/s ed è stato dimostrato che in canali opportuni è possibili raggiungere portate utili dell'ordine di 1 Gbit/s. Nel seguito vengono descritti con più dettaglio i meccanismi sin qui elencati e illustrate alcune possibili estensioni di TCP per migliorarne le prestazioni in reti ad alta velocità.

Unità dati e pacchetto TCP

Le unità di dati dello strato TCP sono dette segmenti. Il segmento è composto da una intestazione e da un campo informativo, che contiene i dati di utente. Il formato del segmento TCP è illustrato nella figura seguente. Ogni riga contiene 32 bit.



- *Source Port* (Porta di Origine, 16 bit): definisce l'indirizzo logico del processo sorgente dei dati;
- *Destination Port* (Porta di Destinazione, 16 bit): definisce l'indirizzo logico del processo destinatario dei dati;
- *Sequence Number* (Numero di Sequenza, 32 bit): numero di sequenza in trasmissione; contiene il numero di sequenza del primo *byte* di dati contenuti nel segmento a partire dall'inizio della sessione TCP (se SN=m ed il segmento contiene n *bytes* il prossimo SN sarà pari a m+n); la numerazione dei segmenti è quindi effettuata non numerando i segmenti stessi (come in X.25), ma gli ottetti in essi contenuti;
- *Acknowledgement Number* (Numero di Riscontro, 32 bit): numero di sequenza in ricezione; nei segmenti in cui il bit ACK, presentato più avanti, è posto al valore binario 1, questo campo contiene il numero di sequenza del prossimo *byte* che il sistema che emette tale segmento si aspetta di ricevere; nel caso di connessioni interattive bi-direzionali si usa quindi il meccanismo denominato addossamento (o *piggybacking*) dei riscontri; ovvero si utilizzano segmenti contenenti dati di utente per inviare i riscontri al trasmettitore senza dover, a tal fine, inviare dei segmenti appositi;
- *Offset* (4 bit): contiene il numero di parole di 32 bit contenute nell'intestazione di TCP; l'intestazione di TCP è sempre costituita da un numero di bit multiplo di 32 (questo campo è necessario poiché il campo *Options* è di dimensioni variabili);
- *Reserved* (6 bit): riservato per usi futuri, per ora contiene degli zeri;
- *Control bit* (6 bit): i bit di controllo sono:
  - URG: viene posto uguale al valore binario 1 quando il campo *Urgent Pointer* (definito in seguito) contiene un valore significativo;
  - ACK: viene posto uguale al valore binario 1 quando il campo *Acknowledgement Number* contiene un valore significativo;

    - PSH: viene posto uguale al valore binario 1 quando l'applicazione esige che i dati forniti vengano trasmessi e consegnati all'applicazione ricevente prescindendo dal riempimento delle memorie allocate fra applicazione e TCP e viceversa (solitamente infatti è il riempimento delle suddette memorie che scandisce la trasmissione e la consegna dei dati);
    - RST: viene posto uguale al valore binario 1 quando un malfunzionamento impone il *reset* della connessione;
    - SYN: viene posto uguale al valore binario 1 solo nel primo segmento inviato durante il *3-way handshaking* (stretta di mano a tre fasi, una fase di sincronizzazione fra le entità TCP);
    - FIN: viene posto uguale al valore binario 1 quando la sorgente ha esaurito i dati da trasmettere.
- *Window* (Finestra, 16 bit): dimensione della finestra; contiene il numero di *bytes* che, a cominciare dal numero contenuto nel campo *Acknowledgement Number*, il destinatario del segmento può inviare al mittente del segmento stesso senza ricevere riscontri;
- *Checksum* (16 bit): contiene l'informazione di controllo che permette all'entità TCP ricevente di verificare la correttezza del segmento ricevuto;
- *Urgent Pointer* (Puntatore Urgente, 16 bit): contiene il numero di sequenza del *byte* che delimita superiormente i dati che devono essere consegnati urgentemente al processo ricevente. Tipicamente sono messaggi di controllo che esulano dalla comunicazione in senso stretto. A tale traffico ci si riferisce di solito con il nome di *out-of-band* (fuori banda);
- *Options* (Opzioni, di lunghezza variabile): sono presenti solo raramente: le più note sono *End of Option List*, *No-operation* e *Maximum Segment Size* (MSS). Ci si soffermerà, in seguito, solo sull'ultima opzione citata; *Padding* (Riempitivo) (di lunghezza variabile): contiene sempre degli zeri. Serve come riempitivo aggiunto per far sì che l'intestazione abbia una lunghezza multipla di 32 bit.

Servizi TCP e TCP port

Il numero di porta è il valore globale e univoco attraverso il quale un programma *client* indirizza un programma *server*; per richiedere un certo servizio, un applicativo *client* deve aprire una connessione con la macchina di destinazione sulla porta *server* che individua quel particolare servizio.

Un *client* FTP, ad esempio, per connettersi ad un *server* FTP, deve conoscere e indicare l'indirizzo IP dell'elaboratore remoto e il numero della porta associata al servizio FTP. Le porte sono individuate da un numero naturale rappresentato con 16 bit.

Questo spazio di numerazione è diviso in due gruppi:

- da 0 a 1023 è lo spazio riservato per le porte privilegiate o *well known ports*, che servono per indirizzare un certo servizio;
- lo spazio da 1024 a 65535 è lasciato libero per le porte utenti, cioè quelle scelte dall'applicativo *client* come porta sorgente.

Nella tabella seguente vengono riportati i numeri di porta di alcuni tra i servizi più noti.

Numero di porta	**Servizio**
0	Riservata
1	TCPMUX *Multiplexor* TCP
5	RJE *Remote Job Entry*
7	*ECHO* Eco
9	*DISCARD* Scarto

11	*USERS* Utenti attivi
13	*DAYTIME* Ora del giorno
15	Programma di stato della rete
17	*QUOTE* Citazione del giorno
19	*CHARGEN* Generatore di caratteri
20	FTP-DATA *File Tranfer Protocol* (dati)
21	FTP *File Tranfer Protocol* (controllo)
23	TELNET Connessione di terminale
25	SMTP *Simple Mail Transport Protocol*
37	*TIME* Tempo
42	*NAMESERVER Server* di nomi dell'*host*
43	*NICNAME* Chi è
53	*DOMAIN Server* di nomi del dominio DNS
77	Qualunque servizio RJE privato
79	*FINGER Finger* (indicatore)
80	HTTP *HyperText Tranfer Protocol*
93	DCP *Device Control Protocol*
95	SUPDUP Protocollo SUPDUP
101	*HOSTNAME Server* di nomi di *host* NIC
102	ISO-TSAP ISO-TSAP
103	X400 Servizio di posta X.400
104	X400SND Invio di posta X.400
110	POP3
111	SUNRPC RPC di *Sun Microsystems*
113	AUTH Servizio di autenticazione
117	UUCP-PATH Servizio di percorso (*path*) UUCP
119	NNTP Protocollo di trasferimento news USENET
123	NTP NTP
129	PWDGEN Protocollo generatore di *password*
139	NETBIOS-SSN Servizio di sessione di NETBIOS
160-223	Riservati

Controllo in TCP - 3-way handshake

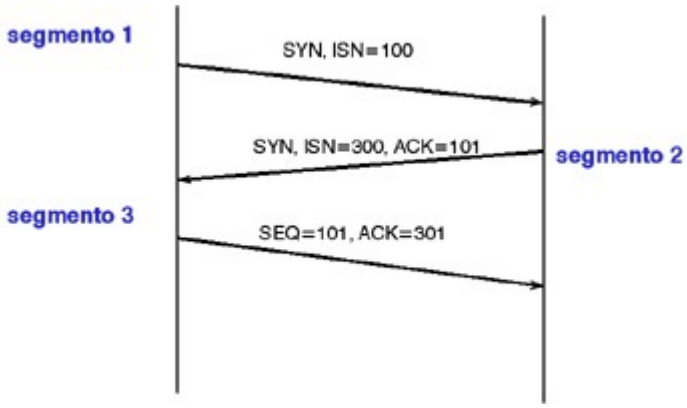
Il protocollo **TCP** è un protocollo orientato alla connessione. Tale affermazione implica che l'entità TCP residente nel sistema mittente deve instaurare una connessione con l'entità TCP residente nel sistema di destinazione, prima che la fase di trasferimento delle informazioni possa avere inizio.

Le due entità TCP interagenti si sincronizzano scambiandosi il proprio numero di sequenza in trasmissione iniziale, che rappresenta il numero a partire dal quale tutti i *byte* trasmessi saranno sequenzialmente numerati una volta instaurata la connessione. All'apertura della connessione, prima dell'effettivo scambio di dati, le macchine coinvolte eseguono una fase di inizializzazione per

scambiarsi il numero di sequenza iniziale, *Initial Sequence Number* (ISN), e per fissare alcuni parametri.

Ogni macchina sceglie il proprio ISN nello spazio a disposizione da 0 a ((2^32) -1); il numero di sequenza in trasmissione non può iniziare da un dato valore fisso; ogni volta che si instaura una nuova connessione si deve scegliere il numero di sequenza in trasmissione da cui iniziare per evitare di usare numeri relativi a vecchie connessioni, fatto che può creare delle sovrapposizioni se alcuni pacchetti, per ritardi della rete, sono ancora in transito sulla connessione.

Tra i *flag* dell'intestazione TCP, il bit SYN attivato indica, che il contenuto del campo *Sequence Number* è valido; il bit ACK indica che il contenuto del campo *acknowledge* è significativo.

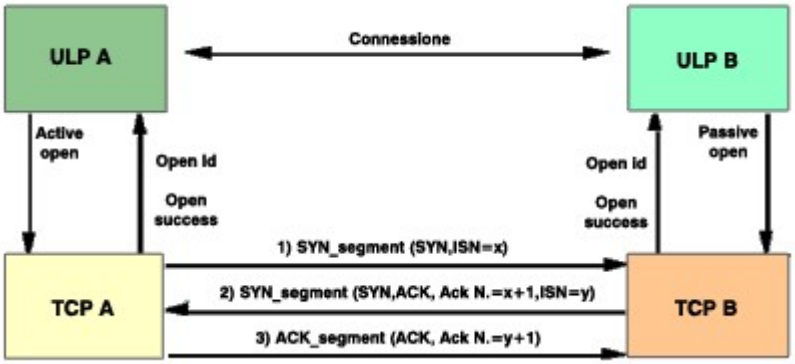


Lo scambio informativo tra le due macchine avviene con i seguenti passi:

1. da A a B: il mio ISN è X (SYN=1;ACK=0);
2. da B ad A: il tuo ISN è X;
3. da B ad A: il mio ISN è Y;
4. da A a B: il tuo ISN è Y (SYN=1;ACK=1).

Gli step 2 e 3 sono combinati in un unico messaggio con i bit SYN=1 e ACK=1; quindi i pacchetti di apertura sono 3.

La figura mostra in maggior campo la procedura *three way handshake*.



Quando deve essere instaurata una connessione allo strato di applicazione fra un dato processo applicativo, denominato ULP A (ULP=*Upper Layer Protocol*), residente nel sistema A, ed un ULP B, residente nel sistema remoto B, il primo passo che si deve compiere è l'invio di una *active open* (primitiva di Richiesta di Servizio) da parte dell'ULP A all'entità TCP A, con la quale quest'ultima viene messa al corrente di tale desiderio.

L'entità TCP A risponde ad ULP A tramite la primitiva *open id*, primitiva di Risposta di Servizio, ed avvia il meccanismo *3-way handshaking* inviando, all'entità TCP ricevente, TCP B, un primo segmento. Tale primo segmento, denominato SYN e contenente il bit SYN posto al valore logico 1, ha un numero di sequenza in trasmissione (denominato *Initial Sequence Number* - ISN), pari al valore assunto dal contatore residente nel sistema A. Per ogni segmento scambiato, vi sono:

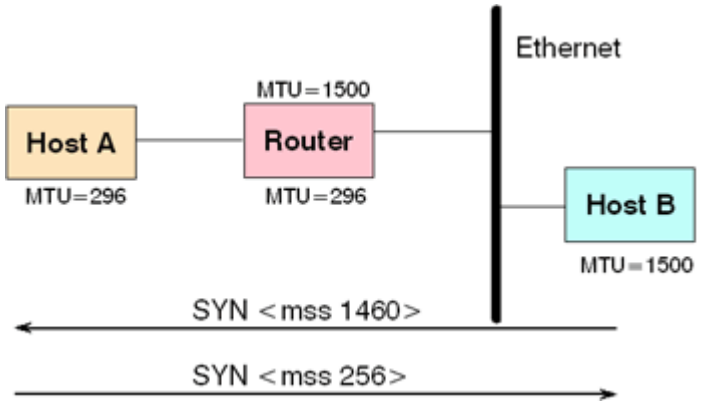
- il primo valore assunto dal campo *Sequence Number*, ISN;
- il valore assunto dal campo *Acknowledgement Number*, Ack. N;
- la scritta SYN e/o ACK quando i relativi bit sono posti al valore logico 1.

Controllo in TCP - MSS

Il *Maximum Segment Size* (MSS) rappresenta la dimensione massima del campo dati che TCP può inviare all'*host* remoto. Quando viene instaurata una connessione, nella fase di apertura, i due *host* annunciano i rispettivi valori di MSS, usando il campo opzioni dell'*header* del TCP. Il valore di *default* è di 536 *byte*. Il datagramma IP risultante è 40 *byte*: 20 *byte* per l'*header* TCP e 20 *byte* per l'*header* IP. Affinché non si abbia frammentazione del datagramma IP deve risultare:

MSS + TCP *header* + IP *header* <= MTU,

dove *Maximum Transfer Unit* (MTU) rappresenta la dimensione massima del pacchetto a livello IP e dipende dalla tecnologia di sottorete. I due *host* si accordano, quindi, per usare il minimo tra i due MSS annunciati; in figura si usano segmenti TCP di 256 *byte*.



Controllo in TCP - Affidabilità

Il TCP garantisce l'affidabilità mediante PAR (*Positive Acknowledge with Retransmission*). Ogni segmento contiene un *checksum* che il ricevitore usa per verificare l'integrità dei dati. Se il segmento è corretto, dal protocollo sulla macchina in ricezione viene inviato un segmento di *acknowledge*. Se il segmento non è corretto, viene semplicemente scartato e non viene inviato nulla alla macchina in trasmissione. Sulla macchina sorgente la mancanza del riscontro, allo scadere di un opportuno *timeout*, indica che il pacchetto è stato scartato o che non è arrivato a destinazione, e che quindi va ritrasmesso.

Il segmento viene costruito cercando di raggiungere la dimensione MSS stabilita al momento della connessione. Ciascun segmento contiene un numero di sequenza che identifica la posizione nel *byte stream* del primo *byte* a partire da ISN. Il riscontro viene inviato su un segmento di *acknowledge* (se è possibile insieme ai dati) con ACK=N, che indica che tutti i *byte* fino ad N-1 sono stati ricevuti correttamente. Insieme all'*acknowledge* viene inviato un valore per la *window*, che definisce il numero di *byte* che il ricevitore è in grado di accettare. Il valore della *window* inviato dal ricevente consente la realizzazione di un controllo di flusso basato sulle necessità della macchina di destinazione.

Controllo in TCP - Finestra

Il trasmettitore determina la sua finestra utilizzabile, che rappresenta quanti dati (*byte*) può trasmettere immediatamente. La finestra si apre man mano che il ricevitore riscontra al trasmettitore i dati inviatigli. Il moto relativo dei due estremi della finestra fa aumentare o diminuire la dimensione della stessa.

La finestra si chiude quando l'estremo sinistro avanza verso destra. Ciò accade quando i dati vengono trasmessi, fino a che gli stessi non sono riscontrati.

La finestra si apre quando l'estremo destro avanza verso destra, consentendo al trasmettitore di inviare ulteriori dati. Ciò accade quando il ricevitore legge i dati riconosciuti, liberando spazio nel suo *buffer* TCP, ed invia i relativi riscontri al trasmettitore.

In questa fase il ricevitore può anche segnalare al trasmettitore una variazione (in più o in meno) della finestra. Se l'estremo sinistro raggiunge quello destro, la finestra è definita zero *window*: il trasmettitore non può inviare dati. In ogni momento, quindi, la finestra di trasmissione fissa il numero di pacchetti che possono essere inviati e, in particolare, la somma dei pacchetti da inviare e di quelli non ancora riscontrati deve essere sempre uguale al valore della finestra.

La finestra di trasmissione è controllata attraverso due meccanismi:

- la finestra annunciata dal ricevitore, che limita la finestra massima di trasmissione;
- la quantità di traffico presente in rete, perché se non ci sono pacchetti riscontrati non se ne possono inviare di nuovi.

In questo modo il TCP si adegua alle condizioni di traffico e trasmette ad una velocità pari alla capacità disponibile in rete.

La finestra di trasmissione viene aumentata per fare in modo che il TCP possa sfruttare tutta la banda disponibile. In partenza la finestra viene aperta con un andamento esponenziale, fino a che non si verifica la prima perdita di un pacchetto; questo evento può essere individuato dalla scadenza di un

timer entro il quale doveva arrivare l'***acknowledge***, o dall'arrivo in trasmissione di *acknowledge* che riscontrano sempre lo stesso pacchetto: in ricezione infatti per ogni pacchetto fuori sequenza si invia un *acknowledge* dell'ultimo pacchetto in sequenza.

Alla prima perdita, la finestra di trasmissione viene ridotta ad un solo pacchetto e si calcola il valore di ssthresh (*slow-start threshold*), pari alla metà del valore che aveva la finestra di trasmissione quando si è verificata la perdita; si ha ancora un andamento esponenziale, fino a quando la finestra assume il valore di ssthresh e da lì in poi la finestra viene aperta in modo lineare, fino alla prossima perdita.

Applicazioni per TCP

Al livello più alto della pila di protocolli si pongono gli applicativi, che possono utilizzare come livello di trasporto UDP o TCP a seconda delle necessità. Poiché UDP è un protocollo di tipo ***connection-less*** (non prevede controllo di flusso o recupero di errore), questo non è consigliabile con applicativi per il trasferimento dati, come **FTP** o **HTTP**, ma può essere utile, grazie al ridotto *overhead*, per applicativi che usano pacchetti di piccole dimensioni, nonché per gli applicativi *real-time* in cui è inutile la ritrasmissione di pacchetti errati.

I più comuni protocolli applicativi sono i seguenti:

- **Telnet**;
- **FTP**;
- **DNS**;
- **Posta elettronica**:
  - formato dei messaggi (RFC 822, *MIME*);
  - trasferimento dei messaggi (SMTP, POP3, IMAP);
- **HTTP**;
- **PROXY**;

Telnet

Telnet è un protocollo che permette ad un utente di collegarsi, tramite elaboratore locale, ad un qualsiasi altro elaboratore remoto connesso alla rete. La connessione viene attivata facendo seguire al comando telnet il nome del calcolatore remoto o il suo indirizzo. Da quel momento in poi, tutti i caratteri battuti sulla tastiera sono inviati all'elaboratore remoto e le risposte da questo generate sono visualizzate sullo schermo locale. Il calcolatore locale è reso trasparente dal programma telnet e si opera come se si fosse direttamente connessi all'elaboratore remoto.

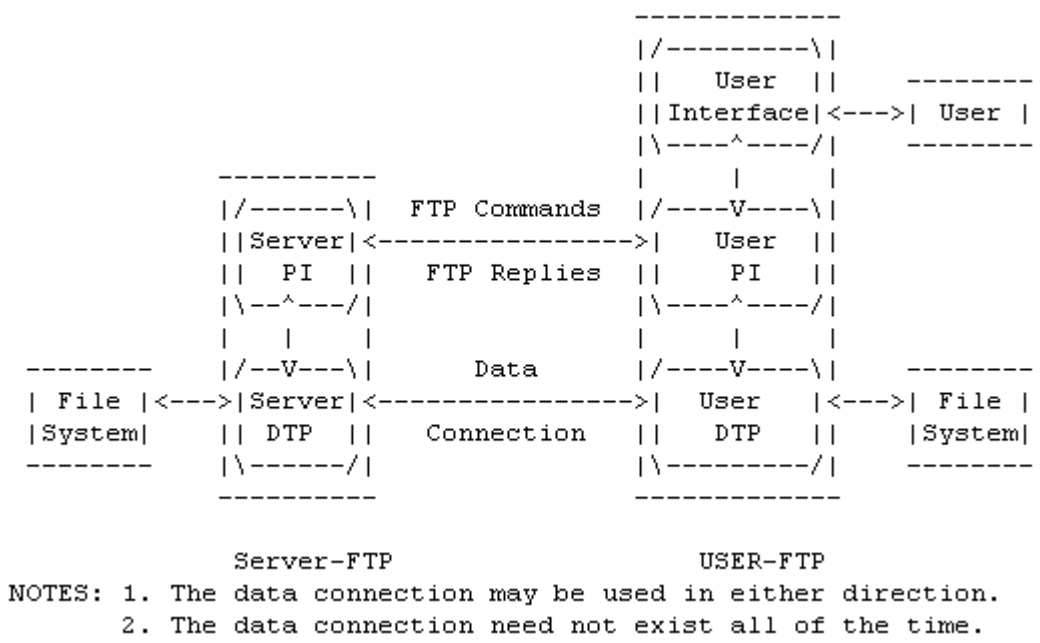
Quando ci si scollega dall'elaboratore remoto, il programma telnet termina e ci si trova nuovamente a dialogare con il sistema operativo dell'elaboratore locale.
Normalmente il programma telnet include degli emulatori per i terminali più diffusi (esempio: Digital VT100 e IBM 3270). Telnet è specificato dalle RFC 854 e 855. Alternativamente al telnet è possibile utilizzare il comando `rlogin` che ha funzionalità analoghe.

FTP

[FTP è specificato nel RFC 959]

Il *File Transfer Protocol* (**FTP**) è una specifica di protocollo applicativo che permette ad un utente collegato ad un calcolatore, di trasferire *file* da e verso un altro elaboratore. La sicurezza è gestita tramite la richiesta all'utente di fornire uno *username* ed una *password* validi presso l'elaboratore remoto.

FTP gestisce la rappresentazione dei dati in maniera automatica di *file* di testo tra elaboratori con codifiche dei caratteri diverse (ad esempio quando si ha a che fare con diversi sistemi operativi e quindi diverse strutture di *file* e diverso set di caratteri). Telnet risolve i problemi di eterogeneità forzando entrambe le macchine a lavorare con uno stesso standard: i caratteri scambiati sono codificati con NVT **ASCII**.



FTP differisce dalle altre applicazioni perché usa due connessioni TCP per trasferire un *file*: una connessione di controllo sulla porta 21, in uso durante tutto il trasferimento del *file* e che serve per passare i comandi del *client* e le risposte del *server* (in particolare per inizializzare il trasferimento di un *file*), e una connessione per i dati, che è creata ogni volta che va trasferito un *file*. FTP supporta un limitato set di tipi di *file* e di strutture di memoria: per trasferire un *file* la macchina *client* e quella *server* devono inizializzarsi e fare delle scelte per le opzioni previste.

Per decidere come un *file* deve essere trasferito e memorizzato, la macchina *client* e quella *server* devono fare una scelta per un formato comune di rappresentazione e trasmissione dei dati ed in particolare:

Tipo di *file*

- *file* **ASCII**, è trasferito con codifica NVT ASCII e necessita per chi trasmette la conversione dal formato locale in ASCII e per chi riceve la conversione opposta; viene inviata la fine di ogni linea, quindi in ricezione si fa una scansione dei *byte* in attesa del *carriage return*; è usato per trasferire *file* di testo;
- *file* immagine o binario, trasferito come un flusso continuo di bit; è usato di solito per trasferire *file* binari;

Struttura

- *file* come flusso continuo di *byte*, senza nessuna struttura interna;
- *file* organizzato con una struttura a record, usato per i *file* di testo;
- *file* organizzato per pagine, in cui si trasmette una pagina alla volta, con numero di pagina che permette al ricevitore di memorizzarle in modo casuale;

Modo di trasmissione

- *stream*, cioé come flusso continuo di bit; per un *file* senza struttura, la fine del *file* viene individuata dalla chiusura della connessione, mentre per un *file* con struttura a record, una speciale sequenza di due *byte* indica la fine dei record e del *file*;
- a blocchi, con *file* trasferito a blocchi ognuno preceduto da un *header*. La combinazione delle precedenti opzioni dà le possibili combinazioni nel trasferimento e memorizzazione dei *file*. La scelta più comune in ambiente *Unix* è come tipo di *file* ASCII o binario, senza struttura e modo di trasmissione *stream*.

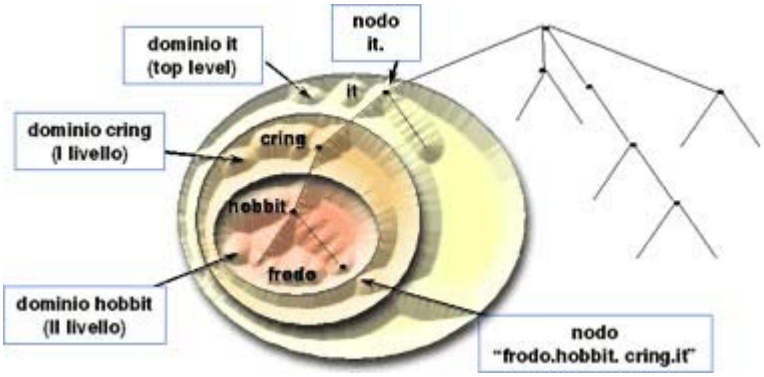
I comandi e le repliche che il *client* e il *server* FTP si scambiano sul canale di controllo sono codificati in NVT ASCII. I comandi che il *client* può inviare al *server* sono circa 30; i più importanti sono riportati nella tabella. Le repliche del *server* sono costituite da un numero di tre cifre usato dal *client* per individuare il tipo di risposta, e da un commento per l'utente. Esempi di *reply* sono: `200 Command OK`, `331 Username OK - password required`, eccetera.

Comando	Descrizione
ABOR	annulla (*abort*) il precedente comando FTP e ogni trasferimento di dati
LIST filelist	elenca (*list*) *file* e *directory*
PASS password	*password* sul *server*
PORT n1, n2, n3, n4, n5, n6	*client IP address* (*nl.n2.n3.n4*) e *port* (*n5 x 256 + n6*)
QUIT	*logoff* dal *server*
RETR filename	ottieni (*retrieve*, *get*) un *file*
STOP filename	immagazzina (*store*, *put*) un *file*
SYST	*server returns system type*
TYPE type	specifica il tipo di *file*: A per ASCII, I per *image*
USER username	*usename* sul *server*

TFTP (*Trivial* FTP) è una versione semplificata di FTP usata normalmente per *downloading* di *software* e specificata nel RFC 1350.

DNS

[Il DNS è specificato negli RFC 1035, 883 e 882]



I programmi raramente si riferiscono agli *host*, alle *mailbox* e ad altre risorse mediante i loro indirizzi di rete IP e tantomeno tramite numeri binari. Sarà preferibile rivolgersi ai sistemi tramite stringhe, come ad esempio utente@dominio. Tuttavia, gli apparati di rete dispongono della sola nozione di indirizzo binario, quindi è necessario un meccanismo che consenta di convertire le

stringhe in indirizzi di rete.

Il *Domain Name Server* (**DNS**) è una base di dati distribuita e replicata per gestire principalmente la corrispondenza tra nomi e indirizzi IP. DNS si avvale di una struttura gerarchica ad albero per stabilire i nomi. La radice è la voce di massimo livello ed è anche il nodo genitore rispetto ai livelli inferiori di un albero. L'albero è costituito da rami, che collegano i nodi.

Le etichette dei nodi dello stesso livello nell'albero devono essere completamente univoche e distinte: ciò significa che l'etichetta deve essere un nome unico e inconfondibile nel livello di nodo specifico.

Ogni dominio, identificato da un nome univoco, controlla ed ha la responsabilità dell'allocazione dei domini nel suo comprensorio. Per creare un nuovo dominio, è necessaria l'autorizzazione del dominio nel quale questo verrà incluso. Una volta che il nuovo dominio è stato creato e registrato, esso può creare a sua volta dei sottodomini senza aver bisogno di richiedere l'autorizzazione a nessuno dei domini superiori.

- **com**: Organizzazioni commerciali (hp.com, *sun*.com ...);
- **edu**: Organizzazioni educative (*berkeley*.edu, *purdue*.edu ...);
- **gov**: Organizzazioni governative (nasa.gov, nsf.gov ...);
- **mil**: Organizzazioni militari (*army*.mil, *navy*.mil ...);
- **net**: Organizzazione di gestione reti (nsf.net ...);
- **org**: Organizzazioni non commerciali (eff.org ...);
- **int**: Organizzazioni internazionali (nato.int ...);
- ***country-code***: Codice di due caratteri per indicare una nazione.

Lo spazio dei nomi è diviso in diversi domini di massimo livello (*top-level domains*), tra i quali distinguiamo dei *top-level domain* generici (*generic domains*) e dei *top-level domain* geografici (*country domains*). Almeno in teoria, un unico *name server* potrebbe contenere l'intero DNS *database* e rispondere a tutte le interrogazioni che lo riguardano. In realtà, questo *server* sarebbe così sovraccarico da essere inutilizzabile. Inoltre, se per qualsiasi motivo questo si guastasse, l'intera Internet si non disporrebbe più del servizio dei nomi.

Non è pensabile quindi che tutte le traduzioni indirizzo IP - *name_address* siano contenute all'interno di un unico *database* o siano decise da una sola organizzazione. Il sistema è organizzato invece con la modalità di *database* distribuito e con il meccanismo della *delegation*: una società o università, proprietaria di una rete, viene delegata per scegliere le traduzioni *IP address - name address* come vuole, e si impegna a mettere a disposizione un *server* DNS che, quando viene interrogato dall'esterno, possa fare queste traduzioni, che quindi sono conosciute solo su quel *server*.

Posta elettronica

[Standard per la formazione di messaggio è specificato nel RFC 822]

La posta elettronica è uno dei servizi più consolidati ed usati nelle reti. In Internet è in uso da circa 20 anni, e prima del WWW era senza dubbio il servizio più utilizzato. Un servizio di posta elettronica, nel suo complesso, consente di effettuare le seguenti operazioni:

- comporre un messaggio;
- spedire il messaggio (a uno o più destinatari);
- ricevere messaggi da altri utenti;
- leggere i messaggi ricevuti;
- stampare, memorizzare, eliminare i messaggi spediti o ricevuti.

Di norma, un messaggio ha un formato ben preciso. In Internet un messaggio ha un formato (definito nell'RFC 822) costituito da un *header* e da un *body*, separati da una linea vuota. L'*header* è a sua volta costituito da una serie di linee, ciascuna relativa a una specifica informazione (identificata da una parola chiave che è la prima sulla linea); alcune informazioni sono:

- ***To***: indirizzo di uno o più destinatari.
- ***From***: indirizzo del mittente.
- **Cc**: indirizzo di uno o più destinatari a cui si invia per conoscenza.
- **Bcc**: *blind* Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
- ***Subject***: argomento del messaggio.
- ***Sender***: chi materialmente effettua l'invio (ad esempio nome della segretaria).

Il *body* contiene il testo del messaggio, in caratteri ASCII. L'ultima riga contiene solo un punto, che identifica la fine del messaggio. Gli indirizzi di posta elettronica in Internet hanno la forma: `username@hostname` dove *username* è una stringa di caratteri che identifica il destinatario, e *hostname* è un nome DNS oppure un indirizzo IP. Ad esempio, frodo@hobbit.cring.it.

La posta elettronica viene implementata in Internet attraverso la cooperazione di due tipi di sottosistemi:

- *Mail User Agent* (MUA);
- *Mail Transport Agent* (**MTA**).

Il primo permette all'utente finale di:

- comporre messaggi;
- consegnarli a un MTA per la trasmissione;
- ricevere e leggere messaggi;
- salvarli o eliminarli.

Il secondo si occupa di:

- trasportare i messaggi sulla rete, fino alla consegna a un MTA di destinazione;
- rispondere ai MUA dei vari utenti per consegnare loro la posta arrivata;
- in questa fase l'MTA richiede ad ogni utente una *password* per consentire l'accesso ai messaggi.

SMTP, POP3 e IMAP

[SMTP è specificato nel RFC 821]
[POP3è specificato nel RFC 1225]

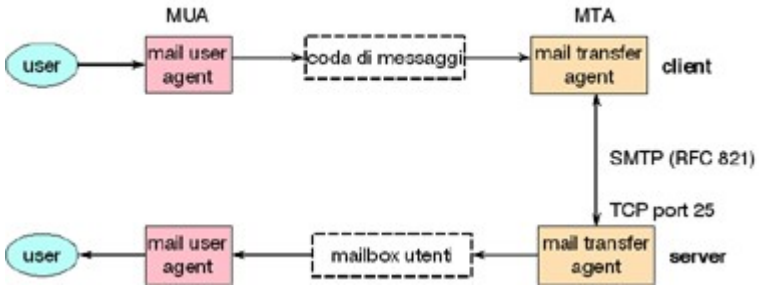
Esiste la definizione di due protocolli per la posta elettronica:

- **SMTP** (*Simple Mail Transfer Protocol*) per il trasporto dei messaggi: dal MUA di origine ad un MTA; fra vari MTA, da quello di partenza fino a quello di destinazione;
- **POP**3 (*Post Office Protocol* versione 3) per la consegna di un messaggio da parte di un MTA al MUA di destinazione.

Il *Simple Mail Transfer Protocol* (SMTP) è probabilmente l'applicativo più importante del TCP/IP. Esso permette di inviare posta elettronica agli utenti della rete.

Ogni utente è identificato dalla sintassi 'Utente@Elaboratore' e non è richiesta alcuna autorizzazione per poter inviare un messaggio di posta elettronica. Il procedimento di invio avviene in *batch*,

riprovando più volte sino a quando l'elaboratore remoto non diventa raggiungibile. L'utente remoto viene avvisato dell'arrivo di un nuovo messaggio. Recentemente sono stati introdotti altri protocolli più sofisticati, quali **IMAP** (*Interactive Mail Acces Protocol*, RFC 1064) e DMSP (*Distributed Mail System Protocol*, RFC 1056), il cui supporto però non è ancora molto diffuso nel *software* disponibile agli utenti.



Inoltre, non è detto che il primo MTA consegni i messagi direttamente all'MTA di destinazine. È possibile che le macchine siano configurate in modo da trasferire i messaggi attraverso un certo numero di *server* SMTP intermedi.

Se un *host*, tipicamente un PC, non è collegato direttamente ad Internet (ad esempio, nel caso di accesso remoto tramite un *Internet provider*), esso utilizza un *mail server* per inviare e ricevere messaggi di posta elettronica. POP3 consente di prelevare i messaggi di posta e memorizzarli sul PC locale, per poi poterli leggere (consultazione *off-line*). Un protocollo più sofisticato è IMAP (*Interactive Mail Access Protocol*). Esso, a differenza di POP3, consente all'utente di leggere i messaggi direttamente dal *server*, senza doverli quindi prelevare e memorizzare sul proprio PC (consultazione *on-line*).

Infine, vanno citate due significative estensioni di funzionalità della posta elettronica:

- possibilità di inviare messaggi di posta contenenti informazioni di qualunque tipo (per esempio programmi eseguibili, immagini, filmati, suoni, eccetera) attraverso lo standard ***MIME*** (*Multipurpose Internet Mail Extension*, RFC 1341 e 1521);
- possibilità di inviare messaggi corredati di firma digitale o crittografati, attraverso lo standard in via di definizione **S/*MIME*** (*Secure/MIME*, RFC 1847).

HTTP

[HTTP 1.0 è specificato nel RFC 1945]
[HTTP 1.1 è specificato nel RFC 2616]

Il protocollo *Hypertext Transfer Protocol* (**HTTP**) è la base del *World Wide Web* (**WWW**). Un *Web client*, chiamato comunemente *browser*, comunica con un *Web server* usando una o più connessioni TCP. La **well-known port per indirizzare sul *server* il servizio è la 80**.

HTTP è il protocollo usato dal *client* e dal *server* per lo scambio di messaggi attraverso connessioni TCP. I documenti che il *server* invia al *client* possono essere immagini, *file* di testo o documenti di tipo HTML (*HyperText Markup Language*).

HTTP è un protocollo semplice: il *client* stabilisce una connessione TCP con il *server* sulla porta 80, fa una richiesta e aspetta il documento di risposta, che in genere contiene puntatori (*hypertext link*) ad altri *file*; questi possono risiedere sullo stesso *server* o su altri. Il *server* indica la fine del documento chiudendo la connessione.

Per poter richiedere un certo servizio da un determinato *host* della rete, si usa un **URL** (*Uniform*

Resource Locator). Il formato per URL è:

```
scheme://hostname[:port]/directory/file
```

dove *scheme* individua il tipo di servizio richiesto, *hostname* è il nome della macchina *server*; segue la *directory* di appartenenza e il **nome** del *file*.

Esempi di schemi sono:

- http: per un *file* su *Web server* (protocollo HTTP);
- ftp: per un *file* su ftp *server* (protocollo FTP);
- telnet: per una connessione su un servizio basato su telnet.

Esempi di URL sono:

- http://info.cern.ch/;
- http://www.ietf.org/;
- ftp://ftp.nis.garr.it/.

Quando un *client* effettua una richiesta invia diverse informazioni:

- il metodo (cioè il comando) che si chiede al *server* di eseguire;
- il numero di versione del protocollo HTTP in uso;
- l'indicazione dell'oggetto al quale applicare il comando;
- varie altre informazioni, fra cui:
  - il tipo di *client*;
  - i tipi di dati che il *client* può accettare.

I metodi definiti in HTTP sono:

- *GET*: richiesta di ricevere un oggetto dal *server*;
- *HEAD*: richiesta di ricevere la sola parte *head* di una pagina HTML;
- *PUT*: richiesta di mandare un oggetto al *server*;
- *POST*: richiesta di appendere sul *server* un oggetto a un altro;
- *DELETE*: richiesta di cancellare sul *server* un oggetto;
- *LINK* e *UNLINK*: richieste di stabilire o eliminare collegamenti fra oggetti del *server*.

In proposito, si noti che il metodo che si usa più frequentemente è *GET*; *POST* ha il suo più significativo utilizzo in relazione all'invio di dati tramite form; *HEAD* si usa quando il *client* vuole avere delle informazioni per decidere se richiedere o no la pagina; *PUT*, *DELETE*, *LINK*, *UNLINK* non sono di norma disponibili per un *client*, tranne che in quei casi in cui l'utente sia abilitato alla configurazione remota (via *Web*) del *server Web*.

Ad esempio, supponiamo che nel *file* HTML visualizzato sul *client* vi sia un'ancora:

```
<A HREF="http://pippo.net/pluto/minnie/index.html"> ..... </A>
```

e che l'utente attivi tale link. A tal punto il *client*:

- chiede al DNS l'indirizzo IP di pippo.net;
- apre una connessione con pippo.net, porta 80;
- invia la sua richiesta.

Essa è costituita da un insieme di comandi (uno per ogni linea di testo) terminati con una linea vuota:

- ***GET* /pluto/minnie/index.html HTTP/1.0**: metodo, URL e versione protocollo;
- ***User-agent*: Mozilla/3.0**: tipo del *client*;
- ***Host*: 160.10.5.43**: indirizzo IP del *client*;
- ***Accept: text/html***: *client* accetta pagine HTML;
- ***Accept: image/gif***: *client* accetta immagini;
- ***Accept: application/octet-stream***: *client* accetta *file* binari qualunque;
- ***If-modified-since*: data e ora**: inviare il documento solo se è più recente della data specificata.

La risposta del *server* è articolata in più parti, poiché il *client* non può sapere in che modo dovrà gestire le informazioni che gli arriveranno. Si consideri ad esempio il fatto che non si può mostrare sotto forma di testo un'immagine o un *file* sonoro, e dunque si deve informare il *client* sulla natura dei dati che gli arriveranno prima di iniziare a spedirglieli.

Per questo motivo la risposta consiste di 3 parti:

- una riga di stato, che indica quale esito ha avuto la richiesta (tutto ok, errore, eccetera);
- delle metainformazioni che descrivono la natura delle informazioni che seguono;
- le informazioni vere e proprie (ossia, l'oggetto richiesto).

La riga di stato, a sua volta, consiste di tre parti:

- Versione del protocollo HTTP;
- Codice numerico di stato;
- Specifica testuale dello stato.

Tipici codici di stato sono:

Esito	Codice numerico	Specifica testuale
Tutto ok	200	*OK*
Documento spostato	301	*Moved permanently*
Richiesta di autenticazione	401	*Unauthorized*
Richiesta di pagamento	402	*Payment required*
Accesso vietato	403	*Forbidden*
Documento non esistente	404	*Not found*
Errore nel *server*	500	*Server error*
SYST	*server returns system type*	*OK*

Dunque, ad esempio, si potrà avere:

```
HTTP/1.0 200 OK
```

Le metainformazioni comunicano al *client* ciò che deve sapere per poter gestire correttamente i dati che riceverà. Sono elencate in linee di testo successive alla riga di stato e terminano con una linea vuota. Tipiche metainformazioni sono:

- ***Server*: ...** : identifica il tipo di *server*;
- ***Date*: ...** : data e ora della risposta;
- ***Content-type*: ...** : tipo dell'oggetto inviato;

- ***Content-length*: ...** : numero di *byte* dell'oggetto inviato;
- ***Content-language*: ...** : linguaggio delle informazioni;
- ***Last-modified*: ...** : data e ora di ultima modifica;
- ***Content-encoding*: ...** : tipo di decodifica per ottenere il *content*;

Il *Content-type* si specifica usando lo standard *MIME* (*Multipurpose Internet Mail Exchange*), nato originariamente per estendere la funzionalità della posta elettronica.

Un tipo *MIME* è specificato da una coppia:

MIME type/MIME subtype

Vari tipi *MIME* sono definiti, e molti altri continuano ad aggiungersi. I più comuni sono:

Type/Subtype	**Estensione**	**Tipologia delle informazioni**
text/plain	.txt, .java	testo
text/html	.html, .htm	pagine HTML
image/gif	.gif	immagini gif
image/jpeg	.jpeg, .jpg	immagini jpeg
audio/basic	.au	suoni
video/mpeg	.mpeg	filmati
application/octet-stream	.class, .cla, .exe	programmi eseguibili
application/postscript	.ps	documenti *Postscript*
x-world/x-vrml	.vrml, .wrl	scenari 3D

Il *server* viene configurato associando alle varie estensioni i corrispondenti tipi *MIME*. Quando gli viene chiesto un *file*, deduce dall'estensione e dalla propria configurazione il tipo *MIME* che deve comunicare al *client*.

Se la corrispondenza non è nota, si usa quella di *default* (tipicamente *text/html*), il che può causare errori in fase di visualizzazione. Anche la configurazione del *client* (in merito alle applicazioni *helper*) si fa sulla base dei tipi *MIME*. Tornando al nostro esempio, una richiesta del *client* quale:

```
GET /products/toasters/index.html HTTP/1.0
User-agent: Mozilla/3.0
eccetera
```

riceverà come risposta dal *server* (supponendo che non ci siano errori) le metainformazioni, poi una riga vuota e quindi il contenuto del documento (in questo caso una pagina HTML costituita di 6528 *byte*):

```
HTTP/1.0 200 OK
Server: NCSA/1.4
Date: Tue, july 4, 1996 19:17:05 GMT
Content-type: text/html
Content-length: 6528
Content-language: en
Last-modified: Mon, july 3, 1996 15:05:35 GMT
_____________________________________<----- notare la riga vuota
<HTML>
<HEAD>
...
```

```
<TITLE>...</TITLE>
...
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

Sulla base di quanto detto finora, si possono fare alcune osservazioni:

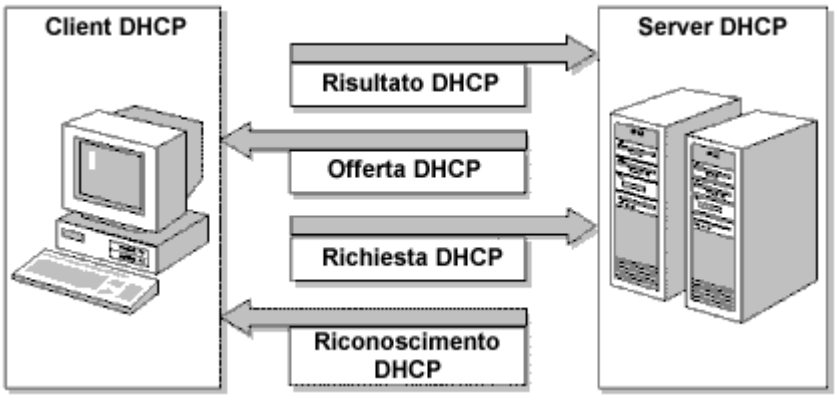
- il protocollo HTTP è molto semplice, essendo basato su interazioni che prevedono esclusivamente l'invio di una singola richiesta e la ricezione della relativa risposta;
- questa semplicità è insieme un punto di forza e di debolezza:
  - di forza perché è facilissimo, attraverso la definizione di nuovi tipi *MIME* e di corrispondenti funzioni sui *client*, estendere le tipologie di informazioni gestibili (il *server* non entra nel merito di ciò che contengono i *file*: si limita a consegnare i dati che gli vengono richiesti, senza preoccuparsi della loro semantica);
  - di debolezza perché queste estensioni di funzionalità talvolta mal si adattano alla concezione originaria (*stateless*) del protocollo, come ad esempio è il caso delle transazioni commerciali.

Dynamic Host Configuration Protocol (DHCP)

DHCP (*Dynamic Host Configuration Protocol*) è standard, secondo quanto definito dalle specifiche RFC 2131 e 2132 IETF. DHCP è in grado di configurare automaticamente un *host* durante il suo avvio su una rete TCP/IP, e può modificare le impostazioni mentre l'*host* è connesso. Ciò consente di memorizzare tutti gli indirizzi IP disponibili insieme alle relative informazioni di configurazione, quali le ***subnet*** *mask*, i **gateway** e gli indirizzi dei *server* DNS, su un *database* centralizzato.

Il protocollo DHCP si basa sul protocollo *BOOTStrap Protocol* (BOOTP), standard Internet (RFC 951 e 1084), che consente l'assegnazione dinamica degli indirizzi IP oltre al riavvio a distanza di *workstation* prive di disco. DHCP supporta l'assegnazione dinamica degli indirizzi IP e fornisce inoltre tutti i dati di configurazione richiesti dai protocolli TCP/IP, più dati aggiuntivi richiesti per *server* specifici.

Come osservato, ciò semplifica i compiti dell'amministratore di rete, al quale adesso sarà sufficiente configurare manualmente solo un *computer*: il *server* DHCP. Quando viene connesso un nuovo *host* sul segmento di rete servito dal *server* DHCP, oppure quando viene riacceso un *host* esistente, il *computer* richiede un indirizzo IP univoco e il *server* DHCP ne assegna uno dal *pool* degli indirizzi IP disponibili.



Questo processo comprende quattro fasi: il *client* DHCP richiede un indirizzo IP (Risultato DHCP), il *server* DHCP offre un indirizzo al *client* DHCP (Offerta DHCP), il *client* DHCP accetta l'offerta e

richiede l'indirizzo (Richiesta DHCP), quindi il *server* DHCP assegna ufficialmente l'indirizzo al *client* DHCP (Riconoscimento DHCP).

Server DHCP

Il *Server* DHCP, attraverso uno strumento di gestione consente agli amministratori di rete di definire le configurazioni dei *client* DHCP. Il *server* DHCP comprende inoltre un *database* per la gestione dell'assegnazione degli indirizzi IP e di altri parametri di configurazione.

I parametri di configurazione **TCP**/IP che possono essere assegnati dal *server* DHCP includono:

- Indirizzi IP di ciascuna scheda di rete dei *computer client*.
- ***Subnet** mask*, che vengono utilizzate per identificare la parte IP della rete dalla parte *host* dell'indirizzo IP.
- **Gateway** predefiniti (*router*), che vengono utilizzati per collegare un singolo segmento della rete agli altri segmenti.
- Parametri di configurazione aggiuntivi che possono essere assegnati ai *client* DHCP, quali ad esempio gli indirizzi IP per i *server* **DNS** o *Windows Internet Naming Service* (*WINS*) che un *client* potrebbe utilizzare.

Client DHCP

Molte piattaforme economiche standard sono in grado di funzionare come *client* DHCP, secondo quanto stabilito dalla specifica RFC 2132 aggiornata.

Le quattro fasi necessarie a un *client* DHCP per ottenere un ***lease*** da un *server* DHCP vengono avviate automaticamente quando il *computer* viene avviato per la prima volta. La configurazione *client* minima richiesta da DHCP può essere abilitata velocemente durante l'installazione del *client* oppure mediante l'esecuzione manuale di una breve reimpostazione delle proprietà TCP/IP del *client*.

Oltre a rendere disponibili le informazioni di configurazione mediante DHCP, gli amministratori di rete sono in grado di sovrascrivere le impostazioni dinamiche mediante impostazioni manuali. Ogni informazione inserita manualmente nella configurazione TCP/IP di un *client* sovrascrive le impostazioni dinamiche.

Per l'esecuzione del proprio lavoro, i protocolli **BOOTP** e DHCP si basano sui *broadcast* di rete. I *router*, in ambienti di *routing* normale, non inoltrano automaticamente *broadcast* da un'interfaccia a un'altra, pertanto, è necessario utilizzare un agente di inoltro che trasmetta tale comunicazione. Un agente di inoltro DHCP può essere un *router* oppure un *computer host* configurato per ascoltare i messaggi **broadcast** DHCP/BOOTP e indirizzarli verso *server* DHCP specifici. L'utilizzo degli agenti di inoltro elimina la necessità di disporre di un *server* DHCP fisico su ogni segmento della rete. Gli agenti di inoltro non soltanto indirizzano le richieste dei *client* DHCP locali ai *server* DHCP remoti, ma restituiscono le risposte dei *server* DHCP remoti ai *client* DHCP.

I *router* conformi alla specifica **RFC** 2131 (che sostituisce la RFC 1542) contengono agenti di inoltro che consentono di inoltrare pacchetti DHCP.

Amministrazione DHCP

Ambiti DHCP

Un ambito DHCP è un raggruppamento amministrativo che identifica gli intervalli consecutivi completi di indirizzi IP possibili per tutti i *client* DHCP su una *subnet* fisica. Gli ambiti definiscono una *subnet* logica destinata a essere fornita di servizi DHCP e consentono al *server* di identificare i

parametri di configurazione che vengono assegnati a tutti i *client* DHCP sulla *subnet*. È necessario definire un ambito prima che i *client* DHCP siano in grado di utilizzare il *server* DHCP per la configurazione TCP/IP dinamica.

***Pool* di indirizzi**

Una volta definito un ambito DHCP e applicati gli intervalli di esclusione, gli indirizzi rimanenti formano all'interno dell'ambito ciò che viene chiamato un *pool* di indirizzi disponibili. Gli indirizzi del *pool* possono quindi venire assegnati dinamicamente ai *client* DHCP sulla rete.

Intervalli di esclusione

Un intervallo di esclusione rappresenta una sequenza limitata di indirizzi IP all'interno di un intervallo di ambito che non devono essere assegnati dal servizio DHCP. Se utilizzati, gli intervalli di esclusione agiscono in modo che ai *client* del *server* DHCP non rilasciato offerto alcun indirizzo contenuto in un determinato intervallo di esclusione.

Prenotazioni

Le prenotazioni consentono al *server* DHCP di assegnare *lease* di indirizzi permanenti. Se utilizzate, le prenotazioni garantiscono la capacità di utilizzare sempre lo stesso indirizzo IP a una periferica *hardware* specifica sulla *subnet*.

Ambiti estesi

È possibile utilizzare una funzionalità amministrativa inclusa nello strumento *Manager DHCP Microsoft* per creare un numero di ambiti separati, raggruppati in una singola entità denominata ambito esteso. Gli ambiti estesi sono utili per risolvere diversi problemi relativi al servizio DHCP.

Lease

Un *lease* rappresenta la durata dell'utilizzo di un indirizzo IP assegnato specificata da un *server* DHCP a un *client*. Quando un *lease* viene rilasciato a un *client*, viene descritto come attivo. A metà della durata del *lease*, è necessario che il *client* rinnovi l'assegnazione del *lease* di indirizzo con il *server*. La durata dei *lease* influisce sulla frequenza delle richieste di rinnovo dei *client* al *server* DHCP relative ai *lease* che sono stati loro assegnati.

Opzioni DHCP

Le opzioni DHCP rappresentano altri parametri di configurazione *client* che un *server* DHCP è in grado di assegnare durante la distribuzione di *lease* ai *client* DHCP. Ad esempio, gli indirizzi IP per un *router* o per un *gateway* predefinito, per i *server WINS* o per i *server* DNS vengono normalmente forniti per un ambito singolo oppure globalmente per tutti gli ambiti gestiti dal *server* DHCP. Molte opzioni DHCP sono predefinite secondo la specifica RFC 2132.