1 Protocolli Internet di livello Applicazione

1.1 I protocolli di Internet

1.1.1 Telnet (port 23, RFC854 STD 8)

Nella classica modalità di collegamento dei terminali ai sistemi centralizzati, i terminali sono collegati all'unità centrale con link fisici da punto a punto, tipicamente attraverso porte seriali RS 232, ed eventualmente attraverso un modem.
Tramite la porta seriale i dati vengono scambiati attraverso un protocollo byte oriented, che specifica, con i caratteri ASCII, cosa il terminale debba visualizzare, quali sono i codici di tastiera e permette di lavorare con modalità "semigrafiche", utilizzando opportune sequenze di caratteri di controllo.
Esistono sequenze che ordinano al terminale di visualizzare caratteri lampeggianti, con scritta e sfondo di diversi colori, che fanno visualizzare caratteri speciali o grafici, e che forniscono varie altre funzioni. Le sequenze sono semplici ma diverse per ciascuno tipo di terminale; ogni tipo di terminale può essere sfruttato al meglio solo usando il suo "linguaggio".
Il protocollo telnet (**tel**ephone **net**work) permette il collegamento con terminali usando TCP/IP, quindi attraverso l'Internet. Usando telnet non c'è più bisogno di un link fisico da punto a punto per collegarsi ad un mainframe, ma basta aprire una sessione TCP, su due host qualunque di una qualunque internet.
Inoltre telnet permette di "astrarre" le funzioni da terminale semigrafico, liberandosi dalla dipendenza dal linguaggio di controllo specifico del terminale utilizzato. Infatti i software telnet, sia dal lato del computer centrale che dal lato del terminale, realizzano una macchina virtuale (NVT, Network Virtual Terminal), con suoi caratteri di controllo e codici di tastiera. Le specifiche della NVT sono usate anche in altri protocolli Internet.
Telnet prevede le presenza di un "server" telnet, che viene eseguito sul computer "centrale", sul quale eseguono i programmi dell'utente, e di un "client" telnet, che viene eseguito sul computer che si deve collegare al computer centrale, e che NON esegue i programmi dell'utente, ma solo il client telnet. Il client dunque lavora come se fosse in emulazione di terminale.
Sia il server che il client telnet lavorano utilizzando il medesimo set di caratteri di controllo, realizzando una sorta di terminale virtuale, il cui linguaggio di comandi è comune fra di loro (vedi figura).
All'inizio del collegamento telnet, client e server realizzano una negoziazione dei parametri che sono in grado di supportare; in seguito, quando il server sa cosa il client è in grado di fare, lo piloterà di conseguenza, ma sempre utilizzando il linguaggio "neutro" di telnet. Sarà poi compito di ciascuno dei due software "tradurre" i comandi "virtuali" telnet nei comandi specifici richiesti dai terminali fisici effettivamente utilizzati.
Tutti i computer che hanno un S.O. di tipo Unix possono fare da server telnet. Per lavorare con quei computer in modalità a carattere è quindi sempre possibile collegarsi tramite Internet usando telnet. Di solito però questa opzione non viene configurata, per problemi di sicurezza.
Anche le ultime versioni dei sistemi "server" di Microsoft possono fare da server telnet, per la configurazione remota del sistema.

Figura: astrazione di terminali con telnet
Siccome telnet prevede che la password sia trasmessa in chiaro, è estremamente vulnerabile allo "sniffing" dei dati. Per questo non è il caso di usarlo in Internet.
Nei sistemi Unix si può usare in alternativa SSH (**S**ecure **Sh**ell, protocollo non internet), che permette di collegarsi a terminale come con telnet, ma che spedisce in modo crittato non solo la password, ma anche tutti i dati.

Client telnet

1.1.2 FTP (File Transfer Protocol, port 20 e 21, RFC959 STD 9))

FTP è il vecchio "cavallo da tiro" di Internet, il protocollo più usato negli anni pioneristici. Viene usato come una shell remota per l'accesso alla struttura delle cartelle ed ai file di un file server internet.
FTP è specificato nella RFC959, ed usa due canali TCP, su due port: un data port (default 20) un control port (default 21). Il client si collega al port 21, client e server si scambiano dati "di controllo" su quel port. L'altro port serve per instaurare una sessione TCP sulla quale si scambiano i contenuti dei file e delle directory (cartelle).

Accesso anonimo
I server FTP presenti sulla Internet hanno di solito "accesso anonimo", cioè concesso a chiunque:
Per entrare su un server FTP anonimo si deve usare il nome di utente "anonymous", la password da usare dipende dal server.

Esempio di login su server FTP:
username: anonymous
```
^^^^ anonymous è l'username più tipico per l'accesso anonimo ^^^^
password: guest
^^^^ Di solito il server viene configurato per accettare la password guest, oppure una password che contenga un "@", e perciò sia un "plausibile" indirizzo di posta elettronica. ^^^^
```

Accesso con password
I server possono riservare alcuni alberi di directory (cartelle) a specifici utenti od a gruppi di utenti. L'accesso a questi file sarà possibile solo dopo aver eseguito l'autenticazione, digitando nome utente e password di un utente abilitato.
FTP non usa password crittate, per cui non è il caso di utilizzarlo in Internet per condividere file "segreti".
Per connessioni sicure ad un server FTP bisogna usare **S-FTP** (secure - FTP), che spedisce sia le password che i file crittati.
FTP, come altri protocolli Internet, si può usare anche "senza client", utilizzando direttamente i suoi comandi. Ciò si potrebbe fare anche da terminale telnet, ma è difficile, dato che bisogna aprire due sessioni contemporaneamente.
Dunque per utilizzare FTP si usa un "FTP client".
Alcuni client FTP permettono di usare in modalità a carattere i comandi specificati dal protocollo FTP.
Un programma "custom", che funzioni a livello di socket. può sfruttare il protocollo in modo completo senza bisogno di ricorrere ad un mailer, utilizzando i comandi che seguono.
In seguito sono presentati i comandi FTP, da spedire sul port di comando (default port 21)

USER <nome d'utente> <CRLF>

PASS <password> <CRLF>

PORT <socket, espresso in byte> <CRLF>
<socket> è la specificazione di una coppia HOST-PORT che si deve usare per stabilire il canale dati.
Esempio:
```
"PORT 196, 132, 28, 1, 2, 3"
```

collega al socket con indirizzo IP 196.132.28.1, al port 515 (256 * 2 + 3, 2 è il byte alto, 3 quello basso)

HELP <CRLF>

LIST [<path o file>] <CRLF>
Se <path o file> specifica una path, il comando LIST è equivalente a dir di MSDOS, se specifica un file verranno mostrate informazioni di dettaglio su quel file.

CWD [<path>] <CRLF>
Change Working Directory: come CD in DOS e Unix

PWD <CRLF>
Print Working Directory: visualizza il percorso (path) della cartella corrente

CDUP <CRLF>
Change Directory "up", alla cartella "madre".

MKD <path> <CRLF>
Make directory: crea una nuova cartella.

RMD <path> <CRLF>
Remove directory: elimina la cartella indicata.

RNFR <nome vecchio del file> <CRLF>
Rename directory FROM: comunica al server il "vecchio" nome di un file, che si vuole cambiare.

RNTO <nome nuovo del file> <CRLF>
Rename directory TO: comunica al server il nuovo nome del file, il cui nome vecchio era era stato spedito precedentemente con il comando **RNFR**. Cambia il nome.

MODE <Modo> <CRLF>
Specifica la modalità del trasferimento dati sul canale dati.
<Modo> = **S** | **B** | **C**
S = Stream: un byte di seguito all'altro, senza distinzione di uno dall'altro
B = Block: trasferimento a blocchi

C = Compressed: dati compressi.

TYPE <Tipo> <CRLF>
Specifica il tipo dei dati.
<Tipo> = **A** | **E** | **I**
A = ASCII: file di tipo testo con codifica ASCII
E = EBCDIC: file di tipo testo con codifica EBCDIC
I = Image: dati binary (es. file .jpg o programma)

RETR <file> <CRLF>
Retrieve: comando di **download**, dal server al client. Recupera dal server il file indicato, che viene spedito nel canale dati.

STOR <file> <CRLF>
Store: comando di **upload**, dal client al server. Memorizza come <file> ciò che il client spedisce nel canale dati.

APPE <file> <CRLF>
Append: comando di upload in "aggiunta": memorizza in fondo al file <file> ciò che il client spedisce nel canale dati . Se <file> non esiste viene creato.

QUIT <CRLF>
Chiude la sessione FTP.

Programmi client
Esistono molti programmi client, in ogni Sistema Operativo, in modalità grafica od a carattere.
Esempio di uso del client FTP in Windows, in modalità a carattere (MSDOS) :

```
C:> FTP
ftp> open megaditta.com <enter>
ftp> 220 <messaggio di saluto>
ftp> user: <nome d'utente>
ftp> password: <password>
ftp> dir
```

Il client Microsoft a caratteri può usare degli "alias" fra il comando "ufficiale" di FTP e quello che viene scritto dall'utente. Per esempio, se si scrive "dir" il client Windows spedisce al server il comando FTP "LIST".
FTP è anche implementato nel Web browser Internet Explorer, basta usare l'URI ftp://<nome di dominio>[<path e file>]. Qualora sia necessario dare una password per accedere al server, bisognerà scriverla (IN CHIARO!) con il nome dell'utente prima del nome di dominio. La password verrà spedita in chiaro sulla rete.

TFTP (RFC1350 STD 33)
Esiste anche una versione "leggera" di FTP, detta "Trivial FTP", che, essendo molto semplice, viene usata in alcuni dispositivi embedded che devono distribuire file (p.es. telecamere Web).

1.1.3 Protocolli per la posta elettronica (e-mail)
Si usa il termine "electronic mail", "e-mail" o "email" per i sistemi che permette la comunicazione fra umani utilizzando messaggi che vengono automaticamente portati a destinazione da una rete di computer.
Nelle comunicazioni per posta elettronica Internet il mittente ed il destinatario non comunicano direttamente, ma sono aiutati da un terzo soggetto, che fa da intermediario e funziona come una cassetta della posta. Questo terzo soggetto è il "server di posta elettronica".
Il mittente si rivolge al server di posta del destinatario e gli spedisce il messaggio. Il server tiene il messaggio nella sua memoria ("mailbox") e lo consegna al destinatario quando questi glielo chiede.
In questo modo il destinatario può benissimo non esistere sulla Internet, ma può ricevere posta comunque.
Per esempio, se il destinatario ha un collegamento Internet su linea commutata esso continua a ricevere messaggi e-mail anche quando non è on-line, perché il suo server di posta, che è collegato ad Internet 24 ore su 24, li tiene da parte per lui, nella sua mailbox.

Dunque per far funzionare la posta elettronica c'è bisogno di "due" client[1], uno in spedizione, dal mittente al server di posta, ed uno in ricezione, dalla mailbox al destinatario. Allo scopo sono stati definiti due distinti protocolli, uno per la spedizione (SMTP) e l'altro per la ricezione (POP3).
Nella quasi totalità dei casi la posta elettronica viene scritta e letta con un unico, apposito, programma client, detto "**mailer",** "mail reader", o "Mail User Agent" (MUA). Esistono molti mailer, i più usati sono: Outlook Express, Mozilla Thunderbird. Eudora.

1 Di solito sono implementati nello stesso programma, il "mailer".

Indirizzi di posta elettronica Internet

ufantozzi@megaditta.com è un indirizzo di posta elettronica. Esso è riconoscibile come tale per via del fatto che è presente il carattere speciale @ ("at"), non usato negli altri protocolli Internet.
A destra di @ c'è un nome di dominio che, come tutti i nomi di dominio, individua un computer in Internet. In quel computer deve essere attivo un server di posta elettronica, che risponde, al port 25, ai comandi del protocollo SMTP.
A sinistra di @ c'è un nome che corrisponde ad un utente definito dall'amministratore di sistema sul server di posta che usiamo. Fra le tante caselle postali che il server SMTP di megaditta.com gestisce c'è anche quella dell'utente "ufantozzi". In Internet i messaggi diretti a ufantozzi non sono spediti direttamente al suo computer (esso potrebbe anche essere spento e, comunque, fuori dall'Internet), ma a megaditta.com, che li memorizza in una sua "casella postale" (mailbox) e li spedisce tutti insieme a ufantozzi, quando questi è collegato e ne fa richiesta.

SMTP (Simple Mail Transfer Protocol, port 25, RFC821 STD 10)

SMTP è il protocollo usato per **spedire** messaggi ad un server di posta elettronica. E' usato anche dai server per spedire i messaggi fra di loro.
SMTP fa uso dei servizi affidabili messi a disposizione da TCP e stabilisce il protocollo attraverso il quale si effettua il trasferimento dei messaggi fra server di posta elettronica. Il client che spedisce la posta elettronica usa SMTP. SMTP è specificato nella RFC821.
SMTP specifica come si trasferiscono i messaggi di posta elettronica, dall'utente al suo server di posta elettronica e da questo server ad altri analoghi. Per questo si definisce il server SMTP come "Message Transfer Agent" (MTA).
Il server SMTP permette anche di conservare in "mailbox" i messaggi ricevuti per conto degli utenti, dalle mailbox l'utente potrà scaricare i messaggi quando lo vorrà.

In SMTP ogni comando è in una singola linea di testo, conclusa dai caratteri "Carriage Return" e "Line Feed" (codici ASCII 13 e 10), che nel seguito indicheremo con <CRLF>.
I comandi SMTP sono composti da quattro caratteri seguiti dai parametri e conclusi da <CRLF>. La conclusione della linea dà attuazione al comando (se non c'è <CRLF> il comando non viene eseguito).
Dopo <CRLF> il server esegue il comando e risponde con un messaggio di acknowledge (approccio tipo "command /reply").
Tutte le repliche del server sono concluse da <CRLF> ed iniziano con un numero, scritto come stringa ASCII in modo testo. Quel numero indica sinteticamente la risposta, con una codifica contenuta nella RFC, e può essere seguito da una stringa che "spiega" agli umani il significato della risposta.
P.es. il mail server potrebbe rispondere così:
"220 mbox.megaditta.com service ready"

Comandi SMTP
Dopo l'instaurazione di un collegamento TCP fra il client ed il server il client spedisce il comando di "saluto":

HELO <dominio del mittente><CRLF>
Quando il server riceve questo comando risponde con una linea che inizia per 220 se tutto va bene, altrimenti comunica un errore con altri codici.
A questo punto il client può già trasferire il messaggio da spedire e lo può fare il due modi.
Il modo più comune, l'unico usato nella Internet, è il metodo MAIL, che fa finire il messaggio nella mailbox dell'utente sul suo server.
Quando l'utente fa uso del S.O. Unix, ed ha aperto una sessione di terminale a carattere si può usare anche il metodo SEND, che spedisce il messaggio direttamente al terminale dell'utente desiderato.
Oltre a MAIL e SEND sono presenti anche i comandi SAML (Send AND Mail) e SOML (Send OR Mail), che permettono di usare sia SEND che MAIL, in base alla circostanza.

MAIL FROM: <<nome del mittente ed indirizzo>**>** <CRLF>
<nome del mittente> indica ciò che l'utente che spedisce il messaggio vuole indicare come suo nome. E' del tutto indicativo e non certifica l'autenticità del nome. In pratica il mittente può mettere ciò che gli pare, il suo nome, uno pseudonimo ("nickname") od anche il nome di un altro. Alcuni server, prima di accettare messaggi, possono fare controlli che il l'"indirizzo" indicati "assomigli" ad un indirizzo di posta elettronica (contenga l'@) o possono verificare se esistono effettivamente in rete l'utente ed il server .

Esempio:
MAIL FROM: Rag.Ugo Fantozzi **<**ufantozzi@megaditta.com**>**

RCPT TO: <indirizzo e-mail del destinatario> <CRLF>
Il "**recip**ien**t**" è il destinatario di messaggio. Possono essere presenti molte linee RCPT, se lo stesso messaggio deve essere spedito a molti destinatari.
Naturalmente se si vuole che il messaggio giunga a destinazione è necessario che l'indirizzo e-mail sia giusto, cioè che il server esista e l'utente indicato sia abilitato.
Esempio:
RCPT TO: pfantozzi@casalinghe.it <CRLF>

A questo punto deve seguire la trasmissione del messaggio, preceduta da un comando che chieda l'autorizzazione a spedirlo:

DATA <CRLF>
A questo comando il server risponde con qualcosa del genere:
250 OK start mail input; end with <CRLF> . <CRLF>
Come il server suggerisce, ora possono seguire i "dati" del messaggio, che devono essere conclusi con una linea composta da un solo punto, cioè da un <CRLF>, un codice ASCII "." ed un altro <CRLF>.

Esempio:
Cara Pina <CRLF>
Anche oggi farò molto tardi al lavoro <CRLF>
. <CRLF>

Ora il messaggio è memorizzato nel server SMTP ed è suo compito operare in modo che esso giunga a destinazione. Se chi spedisce il messaggio non vuole controllare eventuali errori, il suo compito sarebbe finito qui e potrebbe chiudere il socket.
Se il socket viene lasciato aperto, a questo punto il server risponde con:

250 OK

Il client è pronto per spedire un altro messaggio. Se non ha più messaggi da spedire, può concludere il collegamento con un comando QUIT.
Se ci sono errori essi verranno comunicati con codici diversi da 220, che qui trascuriamo.

QUIT <CRLF>
Al quale il server risponde con:
221 Server closing connection

poi il server chiude il suo socket, terminando il collegamento TCP.

SMTP ha pochi altri comandi, fra i quali alcuni permettono di definire ed utilizzare la "lista di distribuzione" del messaggio, alcuni altri comandi sono i seguenti:

HELP <CRLF>
Comando da usarsi per avere una schermata di aiuto sui comandi disponibili; utile solo ai quei pochissimi che scaricano la posta via telnet, e non con un mailer.

RST <CRLF>
Esegue il reset della connessione, con conclusione del collegamento TCP.

VRFY <<username**>>** <CRLF>
Verify chiede di verificare se <username> esiste su quel server.
Alcuni server richiedono che lo username venga compreso fra parentesi angolari, altri no.

Header dei messaggi e-mail
Il formato dei messaggi di posta elettronica è normalizzato nella RFC822 STD 11.
All'interno di ogni messaggio SMTP bisognerebbe scrivere all'inizio una "testata" (header) con alcuni dati relativi al mittente, all'oggetto del messaggio, agli altri indirizzi cui è stato mandato, se è una risposta ad un precedente messaggio e con le altre informazioni di interesse.
Il formato di questa testata non è specificato, ma è diventato di uso comune, così che di solito i client di posta elettronica (mailreader) riescono a mostrare campi omogenei. Peraltro è accaduto che i messaggi scritti con un mail reader non fossero del tutto "leggibili" e manipolabili da un altro reader.
All'header fa seguito il messaggio vero e proprio, che è detto "body" (corpo del messaggio). La testata ed il corpo del messaggio sono separati da una singola linea vuota, nella quale compare solo un <CRLF>.

Esempi di campi tipicamente compresi nell'header del messaggio SMTP:
Date: <data di partenza del messaggio>, dichiarata dal server di partenza e non controllata altrimenti dal server che riceve il messaggio.
Subject: <Oggetto del messaggio>, è ciò che appare nella griglia dei messaggi del mailer come stringa "di titolo" del messaggio.
Re: o R: <stringa> "Reply" (replica, risposta) indica il fatto che è un messaggio scritto come replica ad un precedente messaggio proveniente dall'indirizzo del destinatario.

Cc: (carbon copy, "copia in carta carbone", in Italiano sarebbe "per conoscenza") contiene una lista degli altri indirizzi cui stato spedito questo messaggio "per conoscenza"
Qualora non si voglia includere questi campi nel messaggio, esso verrà spedito comunque. Ovviamente il mail reader "dall'altro lato" non potrà visualizzare i campi che mancano.
I mailer possono aggiungere alcuni campi "non standard", utili per la gestione della posta quando "dall'altra" parte ci sia un reader analogo. Inoltre i relay di posta elettronica che stanno fra il mittente ed il destinatario possono modificare l'header, per lasciar traccia del percorso seguito dal messaggio per arrivare a destinazione.

Si presentano ora alcuni esempi di messaggi "reali" SMTP.
Le righe con la notazione ^^^^ COMMENTO ^^^^ sono commenti dell'autore NON presenti nel messaggio originale.

Il seguente messaggio è stato ricevuto da una mailing list della NASA, che riguarda notizie sulla stazione spaziale internazionale (vedi oltre per le mailing list). Si tratta di un normale messaggio di posta,con qualche campo in più nell'header.

```
Return-Path: <hsfnews-owner@vesuvius.jsc.naso.gov>
^^^^ questo campo indica l'indirizzo di posta del gestore della "mailing list"
^^^^  NON E' PRESENTE NEI CAMPI DELL'HEADER DI UN MESSAGGIO DI POSTA "NORMALE" ^^^^
Received: from vesuvius.jsc.naso.gov ([198.122.144.22]) by fep20-svc.drin.it
          (InterMail vM.4.01.03.13 201-229-121-113) with ESMTP
          id <20010506074011.ILRT18302.fep20-svc.drin.it@vesuvius.jsc.naso.gov>;
          Sun, 6 May 2001 09:40:11 +0200
Received: from localhost (daemon@localhost)
    by vesuvius.jsc.naso.gov (8.11.2/8.11.2) with SMTP id f467dFZ00386;
    Sun, 6 May 2001 02:39:15 -0500 (CDT)
Received: by ginger.jsc.naso.gov (bulk_mailer v1.12); Sun, 6 May 2001 00:44:27 -0500
Received: (from majordom@localhost)
    by vesuvius.jsc.naso.gov (8.11.2/8.11.2) id f465iRq28442;
    Sun, 6 May 2001 00:44:27 -0500 (CDT)
Received: (from www@localhost)
    by vesuvius.jsc.naso.gov (8.11.2/8.11.2) id f465iQF28437
    for hsfnews@listserver.jsc.naso.gov; Sun, 6 May 2001 00:44:26 -0500 (CDT)
^^^^ nella parte iniziale dell'header sono indicati i vari server SMTP che  questo
^^^^ messaggio ha attraversato prima di uscire dalla NASA ^^^^
Date: Sun, 6 May 2001 00:44:26 -0500 (CDT)
^^^^ data scritta "all'americana" ^^^^
Message-Id: <200105060544.f465iQF28437@vesuvius.jsc.naso.gov>
^^^^ identificatore univoco del messaggio sul server di partenza ^^^^
Subject: 2001 International Space Station Status Report #12
^^^^ oggetto del messaggio, verrà scritto dal mail reader fra i dati del messaggio,
^^^^ non all'interno del suo corpo ^^^^
From: info@jsc.naso.gov
To: hsfnews@vesuvius.jsc.naso.gov
^^^^ il messaggio è stato spedito dall'estensore a questo indirizzo, che corrisponde
^^^^ al programma che gestisce la mailing list (vedi oltre), il programma che gestisce la mailing
^^^^ list l'ha poi spedito a tutti i componenti della lista ^^^^
Precedence: bulk
^^^^ urgenza del messaggio (bulk = meno urgente del "normale"), verrà mostrata nel mail reader con
simboli diversi accanto al Subject del messaggio ^^^^
Reply-To: info@jsc.naso.gov
^^^^ campo che indica a chi spedire eventuali "Reply" a questo messaggio ^^^^

^^^^ QUI SOPRA C'E' LA PRIMA RIGA VUOTA, CHE CONCLUDE L'HEADER DEL MESSAGGIO ^^^^
^^^^ il mail reader mostra come messaggio SOLO CIO' CHE SEGUE ^^^^
2001
Report # 12
 1 a.m. CDT, Sunday, May 6, 2001
 Mission Control Center, Houston, Texas

 INTERNATIONAL SPACE STATION STATUS REPORT #12
Expedition Two Crew
Sunday, May 6, 2001 - 1 a.m. CDT


The Soyuz 2 crew successfully undocked
^^^^ qui il messaggio continua (omissis) ^^^^

NASA Johnson Space Center Shuttle Mission/Space Station Status Reports and other
information are available automatically by sending an Internet electronic mail message to
majordomo@listserver.jsc.naso.gov. In the body of the message (not the subject line)
users should type "subscribe hsfnews" (no quotes).
^^^^ qui il messaggio termina ^^^^
```

MIME (RFC2045, 2049, 2046 e 2047)

Gli standard iniziali sulla posta elettronica prevedevano la trasmissione di soli file di testo "puri". In seguito sono state definite delle norme che permettono di "allegare" alla posta uno o più file, eventualmente anche binari, specificati attraverso le regole di **MIME** (Multipurpose Internet Mail Extensions).
MIME è uno standard Internet che stabilisce un formato univoco per dichiarare i diversi tipi di file che possono essere allegati ad un messaggio di posta elettronica o Web (HTTP).
I file binari MIME vengono spediti nello stesso stream TCP in cui vengono spediti anche i dati di testo, vengono perciò "mischiati" dati ASCII e dati binari. I programmi mailer di solito salvano in dati in file così come li ricevono tramite TCP, per cui il messaggio e tutti i suoi file allegati si trovano nello stesso file, separati da stringhe espresse secondo il formato MIME.
I vari tipi di formati MIME sono registrati internazionalmente presso la IANA secondo le specifiche della RFC 2048 (vedi anche "official MIME types" in "ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types")).

Come vedremo in seguito, i documenti MIME vengono usati anche nel protocollo HTTP, per il trasferimento ipertestuale di file multimediali, come grafica, audio e fax.
Il protocollo MIME è stato esteso per garantire connessioni protette, attraverso **S-MIME** (Secure - MIME).
S-MIME permette l'autenticazione dell'utente con firme digitali e la crittazione con chiave pubblica dei messaggi che vengono spediti.

Commentiamo ora un messaggio che comprende degli allegati MIME:

```
From: "FANTOZZI Ugo" <ugo@drin.it>
To: "Ugo FANTOZZI" <ufantozzi@megaditta.it>
Subject: trasmissione antivirus
Date: Wed, 26 Feb 2003 23:37:09 +0100
^^^^ anche se il mailer che ha spedito è "italiano" scrive queste informazioni "all'americana" ^^^^
MIME-Version: 1.0
^^^^ indica lo standard con cui sono rappresentati i file allegati ^^^^
Content-Type: multipart/mixed;
^^^^ indica che nello stesso stream sono presenti più "parti", di formati diversi (mixed) ^^^^
     boundary="----=_NextPart_000_0005_01C2DDEF.FAD85900"
^^^^ dichiarazione dell'indicatore della fine di una "parte" del messaggio ^^^^
^^^^ le "parti" che corrispondono ai file allegati ^^^^
X-Priority: 3
X-MSMail-Priority: Normal
^^^^ priorità, indicata con la convenzione del mailer Microsoft ^^^^
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
^^^^ mailer che ha spedito (della Microsoft) ^^^^
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000

^^^^ LINEA VUOTA: è la fine dell'header del messaggio ^^^^
This is a multi-part message in MIME format.
^^^^ indica che è un messaggio MIME "misto", non è composto solo di testo ^^^^
^^^^ il mailer "interpreta" il contenuto del messaggio che segue e non lo scrive tutto ^^^^
^^^^ nella finestra di testo. Scrive solo le parti che non sono "attachment" (vedi oltre) ^^^^

^^^^ LINEA VUOTA ^^^^
^^^^ ora viene la prima "parte" del messaggio, che viene mostrata nel testo ^^^^
------=_NextPart_000_0005_01C2DDEF.FAD85900
^^^^ inizia una nuova parte del messaggio (è lo stesso separatore definito prima) ^^^^
Content-Type: text/plain;
^^^^ siccome la prima parte è il testo del messaggio, ha tipo MIME  text/plain ^^^^
     charset="iso-8859-1"
Content-Transfer-Encoding: 8bit

^^^^ LINEA VUOTA fine dell'header MIME della prima parte di testo ^^^^
Gentile Sig. FANTOZZI
^^^^ inizio del testo, che poi continua: ^^^^
Le spedisco l'antivirus che mi aveva chiesto (antivirus.zip).
Nel file antivirus.txt trova le istruzioni per l'installazione.

^^^^ LINEA VUOTA che conclude la parte MIME (il testo del messaggio) ^^^^
------=_NextPart_000_0005_01C2DDEF.FAD85900
^^^^ inizio di una nuova parte (file zip) ^^^^
Content-Type: application/x-zip-compressed;
^^^^ tipo MIME dei file zip ^^^^
     name="antivirus.zip"
^^^^ nome della parte MIME ^^^^
Content-Transfer-Encoding: base64
^^^^ formato di codifica dei dati (adatto alla trasmissione di dati binari) ^^^^
Content-Disposition: attachment;
^^^^ indica che è un allegato (attachment), quindi non verrà scritto nella finestra del testo ^^^^
     filename=" antivirus.zip "
^^^^ nome del file, il mailer lo visualizza fra gli allegati, pronto per il click ^^^^
```

```
^^^^ LINEA VUOTA, conclude l'header della seconda parte MIME, segue il file,
^^^^ espresso in forma binaria con codifica base64
UesDBBQAAAAIACB9HikMCRtc0hMAAKBxAAAfAAAAQWNjZXNzIEZpZWxkIE1hcHBlci9mcm1NYWlu
^^^^ primi byte dell'allegato (in forma binaria) ^^^^
FS0AUEsDBBQAAgAIACZTkig3fx0szwAAAB4BAAAKAAAAUmVhZG1lLnR4dE2Pu04DMRBF+0j5h9ul
^^^^ continua con tutto il contenuto binario dell'allegato OMISSIS .. ^^^^

^^^^ LINEA VUOTA che indica la fine dell'allegato ^^^^
----=_NextPart_000_0005_01C2DDEF.FAD85900
^^^^ inizia una nuova parte del messaggio ^^^^
Content-Type: text/plain;
^^^^ la nuova parte del messaggio è un testo (allegato) ^^^^
     name="antivirus.txt"
     charset="iso-8859-1"
Content-Transfer-Encoding: 8bit
Content-Disposition: attachment;
     filename="antivirus.txt"

^^^^ LINEA VUOTA, conclude l'header della seconda parte MIME ^^^^
Istruzioni per l'installazione dell'antivirus
^^^^ omissis (completamento del file di testo  antivirus.txt) ^^^^

^^^^ LINEA VUOTA ^^^^
------=_NextPart_000_0005_01C2DDEF.FAD85900--
^^^^ conclusione dell'ultima parte ^^^^

^^^^ LINEA VUOTA cui non segue nient'altro: conclude tutto il messaggio^^^^
```

POP3 (port 110, RFC1939 STD 53)

POP (Post Office Protocol = protocollo dell'Uffico Postale), versione 3 è il protocollo tramite il quale un client può scaricare i messaggi che si sono accumulati nella sua mailbox su un server SMTP sul quale sia accreditato come utente. Il protocollo è specificato nella RFC1725.
Dopo che il collegamento TCP sul port 110 è attivo, il server inizia il colloquio.
Anche POP ha un funzionamento di tipo comando / replica, simile a SMTP. La replica del server inizia sempre con "+" o con "-". Il "+" indica che tutto è andato bene, "-" che c'è stato un errore.
Al collegamento POP3 è in uno stato detto "Authorization"; il protocollo ha tre stati, in ciascuno di essi è permesso solo un sottoinsieme di tutti i comandi POP3.
Gli stati sono:

Authorization	la condizione del collegamento prima dell'autenticazione dell'utente
Transaction	la condizione in cui il collegamento "funziona"
Update	la condizione in cui si sta chiudendo il collegamento e si rilasciano le risorse

La prima che cosa deve accadere è l'autenticazione dell'utente, per cui i comandi permessi in stato "Authorization" hanno a che fare esclusivamente con la spedizione e la verifica della coppia nome utente – password.

La versione 3 POP permette di fare l'autenticazione spedendo la password in modo critttato, per evitare la sua lettura se i datagrammi IP vengono intercettati.

Vediamo l'inizio di una sessione TCP che scarica un messaggio via POP, con spedizione non crittata della password; all'inizio della linea è indicato in colore chi spedisce il messaggio che segue:

```
<Server>:    +OK POP3 server ready <posta_inferiori@megaditta.com>
^^^^ il server si presenta ^^^^
<Client>:    USER ufantozzi
^^^^ l'utente si presenta ^^^^
<Server>:    +OK Password required for transaction
^^^^ il server vuole una password ^^^^
<Client>:    PASS silvani
^^^^ una password sicura! ^^^^
<Server>:    +OK ufantozzi has 2 message(s)
^^^^ l'utente ha 2 messaggi nella sua mailbox ^^^^
<Client>:    <uno dei comandi successivi>
```

Una volta autenticato l'utente, si passa in stato "transaction"; i più importanti comandi che si possono spedire in questo stato sono i seguenti:

LIST [<n del messaggio>] <CRLF>
Dà la lunghezza in Byte del messaggio indicato (se non si indica alcun messaggio vengono listati tutti i messaggi).

TOP <n del messaggio>**,** <n linee del corpo> <CRLF>
Dà l'header del messaggio <n del messaggio> e le prime <n linee del corpo> del suo corpo

se <n linee del corpo> = 0 non voglio avere il testo del messaggio, ma solo titolo e header "di sistema" (che è diverso dalla parte del messaggio che sta al suo inizio e che potremmo chiamare header "di messaggio").
Naturalmente questo comando si può dare solo in stato "Transaction".
es.
TOP 1, 0
dà le linee dell'header del messaggio 1

RETR <n del messaggio>**,** <n linee del corpo> <CRLF>
Retrieve: recupera il messaggio dalla mailbox corrente (SOLO il corpo del messaggio)

IMAP (RFC2060)
Il protocollo IMAP (Internet Message Access Protocol), viene usato al posto di POP3, con funzioni più sofisticate, e permette ad un client di accedere in remoto alla posta in un server e di manipolare le mailbox, come se esse fossero locali.
IMAP prevede funzioni per creare, cancellare e rinominare le mailbox ed i messaggi, verificare se ci sono nuovi messaggi, leggerli, cercarli.
Inoltre permette rendere pubbliche le mailbox e condividerne il contenuto fra più utenti, divenendo in questo modo uno strumento per la collaborazione di gruppo.
IMAP non specifica come spedire la posta, che è ancora compito di SMTP, ma come manipolarla "in lettura". E' specificato in RFC2060 e RFC2061.
Tutti i mailer odierni ed i più usati server si posta elettronica supportano IMAP, che però, a dispetto della sua utilità, viene ancora poco usato.

1.1.4 Strumenti per la comunicazione di gruppo

Newsgroup NNTP (Network News Transfer Protocol, port 11, RFC977 no STD)
Un newsgroup è un archivio distribuito di messaggi "text based", del tutto analoghi a quelli di SMTP, che vengono replicati in rete fra diversi server, in modo che in ciascuno dei server risiedano copie identiche degli stessi documenti.
Il protocollo NNTP (RFC977) stabilisce le funzioni che i server ed i client NTTP devono realizzare. Il protocollo viene usato anche fra server, per la "replica" dei file.
I messaggi NNTP, che si chiamano "**articoli**" (articles) o "**post**", vengono indirizzati a "**newsgroup**" (gruppi di discussione).
Accedendo ad un server per newsgroup (NNTP) si possono scaricare tutti i messaggi che transitano su tutti i gruppi di discussione che quel server fa circolare.
I nomi dei newsgroup dicono già molto del contenuto. Hanno una struttura gerarchica che individua quello che dovrebbe essere l'argomento dei messaggi che vi sono "contenuti".
Gli argomenti di discussione sono indicati "da sinistra a destra" con nomi simbolici separati dal punto. Andando da sinistra a destra gli argomenti di discussione sono via via "ristretti".
Es.
it.istruzione.scuola, oppure it.istruzione.scuola.informatica

Il nome più a sinistra (it nell'esempio) è detto "gerarchia" (jerarchy) e dà l'indicazione dell'autorità che gestisce tutti i newsgroup il cui nome inizia con quel simbolo. Se si vuole istituire un newsgroup ci si deve rivolgere all'autorità che gestisce la relativa gerarchia.

Per spedire un messaggio rivolto ad uno specifico gruppo di discussione basterà rivolgersi al proprio server, che provvederà a girarlo a tutti gli altri server che ospitano quel gruppo.
La rete dei server NNTP collegati insieme in Internet è detta "**Usenet**". Usenet è una rete di migliaia e migliaia di server dislocati in tutto il mondo, che usa l'infrastruttura di Internet. Tipicamente esiste almeno un server Usenet per ogni provider Internet.

P. es. rec.autos.sport.f1 fa parte della gerarchia "rec" (recreational), che discute di argomenti relativi al tempo libero. In questo gruppo si tratterà di auto, sport, ma in particolare di formula 1. La maggior parte dei gruppi è in inglese. I gruppi della gerarchia "alt" (alternative) sono luogo di discussioni "informali", mentre in "soc" (society) dovrebbero vedersi discussioni più elaborate e in "talk" più chiacchiere. Naturalmente ci sono centinaia di gruppi nella gerarchia "comp" (computer). Per i temi della scienza in generale c'è "sci".
Per i gruppi in cui la lingua di elezione è l'italiano ci sono la gerarchia "it" ed altre gerarchie che fanno riferimento ai maggiori provider italiani. Come detto, il nome dice molto: p.es. comp.lang.basic.visual è un gruppo sul Visual BASIC in inglese e tratta di problemi generali, mentre comp.lang.basic.visual.database è relativo all'accesso a database tramite VB e it.comp.lang.visual-basic è generale ed ha l'italiano come linguaggio di elezione.
Esistono anche gruppi moderati, nei quali una persona incaricata riceve tutti i messaggi diretti al gruppo e decide quali possono "passare" e quali verranno bloccati. In questo modo si può evitare di avere in un gruppo messaggi maleducati o smaccatamente pubblicitari (fenomeno che viene detto "spamming").

Le gerarchie più importanti della usenet sono:
alt, misc ("miscellaneous") , net , rec, free, che ospitano gruppi che trattano tutti i temi, in Inglese
comp, per gruppi relativi all'informatica, in Inglese
sci, per gruppi relativi alla scienza, in Inglese
soc, per gruppi che trattano temi sociali, in Inglese
gerarchie "nazionali" (es. it = Italia, de = Germania), con i nomi di gerarchia a due lettere che corrispondono alla nazione. In questi gruppi le discussioni si tengono tipicamente nella lingua nazionale.
fido, la vecchia "fido network", che esisteva ancora prima di Internet su computer collegati via modem e gestiti da appassionati BBS (Bullettin Board System) e che ora si è "trasferita" in Internet
gnu, gerarchia della "Free Software Foundation"
linux -
Anche le aziende possono permettere che alcuni suoi newsgroup possano essere distribuiti da altri server; p. es. le gerarchie "microsoft" o "netscape" sono distribuite da molti server in rete, ma sono gestite dalle rispettive aziende.

I server della Usenet sono collegati fra loro 24 ore su 24 e devono "sottoscrivere" specifici newsgroup per poter partecipare allo scambio degli articoli. Ogni server è collegato con tre, quattro (o più!) altri server.
Un grosso server Usenet può contenere i messaggi di molte migliaia di gruppi.
Alcuni server NNTP sono in Internet ma non sono nella Usenet, cioè non propagano i loro messaggi attraverso altri server. Questi server funzionano nello stesso modo di tutti gli altri ma per avere accesso ai messaggi che contengono è necessario collegarsi direttamente con essi, non con il più "vicino" della Usenet.

Anche con NNTP ci si potrebbe collegare ai server usando telnet, iniziando un collegamento TCP al port 110 e seguendo attentamente tutte le specifiche del protocollo.
Dunque in linea di principio basta telnet per accedere ai newsgroup. In verità in questo modo le operazioni sono scomode, e per semplificare la vita esistono programmi specifici che danno alcuni vantaggi ulteriori. I programmi client per NNTP si chiamano **newsreader**. I mailer di uso comune sono in grado di fare anche le funzioni del newsreader.
Il newsreader è configurato in modo che sappia qual è l'indirizzo IP o il nome di dominio del server che deve usare.
In alternativa si può usare TCP e scrivere un programma che usi i comandi specificati dalla RFC997.

Il protocollo
I messaggi NNTP devono essere identificati in modo univoco. L'identificatore unico è detto "MessageID", che è affidato al singolo server, che deve scegliere un identificativo univoco. Per l'univocità esso solitamente è nella forma:
<numero>@<nome di domino>.
Il numero di solito dipende dal system time del computer, che lo rende facilmente univoco all'interno del server; il <nome di domino> è "per definizione" un nome univoco, dato che viene assegnato dalla IANA.
Un altro numero che identifica gli articoli è il "message number", che, per lo stesso messaggio è diverso in ogni server.

Le risposte del server hanno un formato che inizia con tre cifre, che chiameremo xyz.
x la prima cifra della risposta esprime la sua condizione relativa agli errori, e viene codificata in questo modo:
- 1 messaggio "informativo", può essere ignorato ai fini della determinazione dell'errore
- 2 OK
- 3 fino a qui OK, manca il resto
- 4 comando giusto, non eseguito per qualche ragione
- 5 comando sbagliato o inesistente

y la seconda cifra è la categoria della risposta:
- 0 messaggi vari
- 1 selezione del newsgroup
- 2 selezione dell'articolo
- 3 funzioni relative alla distribuzione dei messaggi
- 4 posting
- 8 estensioni non standard
- 9 debugging

z è un numero d'ordine che distingue i comandi che hanno gli altri numeri uguali

Formato dell'articolo
Ciò che viene passato fra i server NNTP comprende, all'inizio, un header prodotto in parte dai server ed in parte dall'utente che ha spedito il messaggio (o meglio, dal suo newsreader). Esso dà informazioni sul messaggio, sulla sua identificazione, sul percorso che ha fatto, l'utente che lo ha spedito, il programma che l'ha generato.

Le righe con la notazione ^^^^ COMMENTO ^^^^ sono commenti dell'autore che non compaiono nel messaggio trasferito fra i server.

```
Path: news1.drin.it!news-out.drin.it!news-
in.drin.it!area.cu.mi.it!draco.megaditta.com!not-for-mail
```

```
^^^^ percorso nella "catena" dei newsserver Usenet !^^^^
From: "Ugo Fantozzi" <ufantozzi@megaditta.com>
^^^^ "dichiarazione" di nome ed e-mail del mittente !^^^^
Newsgroups: alt.fan.cicciolina
^^^^ gruppo cui si spedisce l'articolo !^^^^
Subject: Re: R: R: speditemi altre foto
^^^^ oggetto dell'articolo !^^^^
Date: Fri, 27 Apr 2001 23:35:19 GMT
^^^^ "dichiarazione" della data e dell'ora della partenza !^^^^
Organization: megaditta.com
^^^^ provider del newsserver di partenza (aggiunto dal server) !^^^^
Lines: 2
^^^^ numero di linee di messaggio (header escluso) !^^^^
Message-ID: <3aeB020a.5241A840@news.megaditta.com>
^^^^ Message-ID !^^^^
References: <3aDd2123.40264C10@news.megaditta.com>
<AalG6.9504$sb5.1182563@news.autostrada.it> <3ae9e429.44767789@news.megaditta.com> <Lbm-
G6.9951$sb5.1281279@news.autostrada.it>
^^^^ indicazione dei messaggi precedenti sullo stesso argomento; con queste informazioni
^^^^ il newsreader può ricostruire il "thread" del messaggio !^^^^
NNTP-Posting-Host: pi1-379.dialup.megaditta.com
X-Trace: pegasus.megaditta.com 988414534 25417 62.11.1.123 (27 Apr 2001 23:35:34 GMT)
X-Complaints-To: newsadmin@tiscali.it
NNTP-Posting-Date: 27 Apr 2001 23:35:34 GMT
^^^^ effettiva data di spedizione del messaggio al server (aggiunto dal server) ^^^^
X-Newsreader: Forte Free Agent 1.11/32.235
^^^^ newsreader utilizzato (aggiunto dal newsreader) ^^^^
Xref: news-in.drin.it it.fan.cicciolina:5131616
X-Received-Date: Sat, 28 Apr 2001 01:34:02 MET DST (news1.drin.it)

^^^^ LINEA VUOTA per la fine dell'Header ^^^^
Quelle di ieri mi erano piaciute molto!

^^^^ il testo dell'articolo, è concluso da  una linea vuota ^^^^
```

In alcuni casi ci può essere un header "più vecchio", nel quale ci sono meno informazioni ed ove invece d i "Subject" sta scritto "Title".

Formato dei comandi
I comandi NNTP sono costituiti di una sola parola, alcuni hanno parametri; i parametri sono separati da " " e Tab e terminati con i due caratteri "Carriage Return" e "Line Feed". Di seguito elenchiamo i principali comandi NNTP:

HELP <CRLF>
Dà una lista dei comandi disponibili sul server.

LIST <CRLF>
Dà una lista di tutti i newsgroup presenti in un server.

GROUP <nome del gruppo> <CRLF>
Seleziona il newsgroup corrente. Con il messaggio di ritorno il server indica il numero di messaggi del gruppo presenti.

LISTGROUP [<nome del gruppo>] <CRLF>
Dà una lista dei numeri di articolo di tutti gli articoli del newsgroup indicato. Se il nome del gruppo non è indicato usa quello precedentemente selezionato con GROUP.

ARTICLE [<n articolo | Message ID>] <CRLF>
Chiede header e testo di uno specifico articolo del newsgroup corrente.

NEWNEWS <nome del gruppo> <data> <CRLF>
Legge tutti gli articoli presenti sul server che sono stati creati dopo la data specificata . La data deve essere espressa nel formato yymmdd hhmmss, esempio 011231 235959 corrisponde alle ore 23:59:59 del giorno 31 Dicembre 2001.

POST <CRLF>
Chiede il permesso di scrivere un messaggio nel newsgroup corrente.
Se la risposta è 440 = NO non si può scrivere il messaggio; se invece è 340 = OK esso si può scrivere.
L'inizio del messaggio deve comprendere tre linee, seguite dal testo:

```
From <indirizzo di posta elettronica>
Newsgroup <nome del gruppo in cui "postare">
Subject <oggetto del messaggio (titolo riepilogativo)>
LINEA VUOTA
    <testo dell'articolo>
LINEA VUOTA
```

La linea vuota finale conclude il messaggio.

IHAVE <Message ID> <CRLF>
Messaggio spedito da un server ad un altro server per informarlo che ha un nuovo messaggio da "propagare".

Il protocollo NNTP prevede anche la possibilità di ricercare nomi di gruppi e articoli che corrispondono ad un certo schema, detto WILDMAT ("**Wild**card **Mat**ch"), una versione semplificata delle "regular expression" Unix.

Uso del newsreader
La prima volta che un newsreader si collega ad un news server esso scaricare tutti i titoli dei gruppi che il server contiene, che possono essere diverse migliaia (practice patience!). Fra i nomi dei gruppi scaricati si potrà scegliere a quali "abbonarsi" (subscrobe).
Di solito i newsreader presentano una vista ad albero dei gruppi a cui ci si "abbona" (subscribe). La vista parte dal nome del server ed ha, nei livelli successivi, i nomi dei gruppi scelti ed titoli dei messaggi presenti attualmente sul server.
La maggior parte dei newsreader permette di memorizzare i messaggi scaricati e di leggerli "offline".
Ogni server ha un database dei gruppi di discussione che "propaga" nella Usenet. Questo database viene spedito dietro richiesta a chi sia interessato.
Se si vuole leggere il contenuto del messaggio si fanno due clic sul suo titolo. In alternativa si possono "segnare" tutti i titoli che interessano (usando il tasto destro del mouse), per fare in modo che il loro contenuto venga scaricato tutto insieme, così da poterli leggere offline.
L'"iscrizione" ad un newsgroup e' "locale" al computer client. Il programma che legge le news si ricorda i gruppi cui l'utente è "abbonato" e li include nell'albero in cui si possono vedere i titoli dei messaggi.
Il news server non mantiene particolari informazioni sull'utente. Spedisce solo i testi che gli vengono richiesti dal newsreader e spedisce agli altri server con cui è in contatto i messaggi che l'utente vuole mandare a tutto il gruppo. Di conseguenza anche la cancellazione di un "abbonamento" è affare del reader e non del server.
Normalmente il reader legge solo gli header dei messaggi e ne scarica il testo solo su richiesta dell'utente. Se l'utente è interessato regolarmente alla maggior parte dei messaggi scambiati in un gruppo può configurare il reader per far scaricare automaticamente tutti i messaggi che non ha già letto (comando NEWNEWS). Si può scegliere di avere tutto, "body" compreso, oppure solo i titoli.
Il messaggio che riguarda un argomento e tutta la sequenza delle risposte che riguardano quell'argomento (hanno lo stesso "Subject") viene detta "thread". Il newsreader mostra gli articoli dello stesso dell'argomento ("thread") in una struttura ad albero.
I vari newsreader hanno altre funzioni, alcune potenti. Per esempio si può:
- mettere filtri (isolare in cartelle particolari o non avere del tutto i messaggi spediti da uno che ti è antipatico o che contengono parole particolari)
- collegarsi a più di un newsserver
- scegliere i gruppi cui abbonarsi con una funzione potente di filtro sui nomi
- comunicare all'utente il fatto che il server cui si è collegato ha istituito nuovi gruppi

I newsgroup hanno regole scritte e non scritte che dovrebbero essere rispettate. Le regole scritte sono riassunte nel "manifesto" del gruppo, che viene regolarmente pubblicato come "post", quelle non scritte si devono imparare "ascoltando" senza scrivere articoli per un po' di tempo; ci sono anche FAQ che spiegano la "netiquette".
Di solito non è ben visto lo scrivere nei newsgroup in formato HTML perché chi ha un reader che non lo supporta si lamenterà pesantemente (ma ormai sono pochi ..).

Costituzione di un gruppo it
Nella gerarchia "it" per fare un nuovo gruppo almeno uno dei partecipanti ad un gruppo esistente deve scrivere un documento, detto "charter", che spiega come si dovrà chiamare il nuovo gruppo, quali sono gli argomenti dei quali dovrà discutere e quali sono le regole di buona educazione da mantenere per essere ben accetti in quel gruppo. Poi spedirà un post con il charter a quei gruppi che possono essere interessati agli argomenti indicati. Da quel momento inizia una procedura di votazione. Ogni utente interessato a quel tipo di discussione dovrà spedire una mail all'organizzazione responsabile per la gestione del gruppo. Se ci sarà un numero sufficiente di interessati entro un certo tempo limite, verrà istituito il nuovo gruppo ed il "charter" ne diverrà il "manifesto".
Per il contenuto dei gruppi italiani si può vedere in http://www.mailgate.org/mailgate/index.html. Questo sito contiene anche un indice full text di tutti gli articoli di tutti i gruppi della gerarchia, con il quale si possono rivedere le discussioni sugli argomenti che interessano.

Per i gruppi internazionali esisteva http://www.dejanews.com/ ora acquisito da Google (si chiama "Google Groups").

Newsgroup "formattatati" in una Intranet
Come strumenti per la comunicazione di gruppo in ambito aziendale sono molto diffusi sistemi proprietari, quali Lotus Domino o MS Exchange ma molti usano NTTP, che è uno strumento più rudimentale ma, accoppiato ad un newsreader che possa leggere HTML, è in grado di trasmettere e mantenere i messaggi fra gli utenti in modo strutturato, organizzati in argomenti che si possono rileggere e ricercare quando servono e possono avere un contenuto multimediale, con l'in-

serzione di grafica e di memo audio e video. Inoltre l'accesso ad un server NNTP può essere protetto da password, garantendo la riservatezza delle comunicazioni.

Mailing list
La mailing list funziona solo con la posta elettronica e non ha un protocollo apposito come per le news. Come dice il nome una mailing list è una lista di indirizzi e-mail gestita automaticamente da un programma apposito (un "demone" che funziona in background su un normale server per posta elettronica SMTP). Ogni volta che il server SMTP riceve un messaggio diretto alla lista esso viene preso dal demone, ricopiato in una e-mail diversa per ogni partecipante alla lista e spedito ad ogni partecipante per il normale canale di posta elettronica. In questo modo ogni partecipante alla lista riceve una copia di tutti i messaggi che ad essa sono stati inviati.
Configurando un filtro di posta elettronica il mail reader potrà isolare tutti i messaggi provenienti dalla lista e metterli tutti in una stessa cartella, in modo che non siano mescolati agli altri normali messaggi e-mail.
La mailing list è molto più riservata di un newsgroup, se il gestore di una lista vuole escludere un elemento che non si comporta correttamente non ha altro da fare che toglierlo dalla lista, per cui le discussioni di solito sono meno "rumorose".
Per iscriversi ad una mailing list di solito si deve spedire un messaggio e-mail al gestore. Questo messaggio, intercettato dal "demone" che gestisce la lista, dovrà contenere una qualche forma convenzionale di dichiarazione della volontà di partecipare alla lista. Di solito si tratta solo di scrivere "subscribe <nome della lista>" nel "subject" del messaggio, lasciando il "body" vuoto (o viceversa). Se il demone trova la scritta "subscribe", inserisce nella lista l'indirizzo di chi ha spedito il messaggio, che trova nel campo "From" della e-mail ricevuta.
Al contrario del newsgroup l'abbonamento ad una mailing list sta perciò sul server e non è locale. Per cancellarsi dalla lista sarà necessario rispettare le istruzioni che si troveranno nel primo messaggio che il demone della mailing list ci manderà (per questo sarà meglio conservare quel messaggio!). Di solito basta mandare all'indirizzo della lista una mail con "unsubscribe <nome della lista>" nel "subject" o nel "body".
Il programma che gestisce una mailing list viene detto "list server". Il list server "classico" su Unix è "majordomo".

Discussioni Online (chat)
Sono strumenti per la comunicazione in tempo reale e vanno dal semplice "chat" testuale alla videoconferenza.
Le chat sono discussioni nelle quali le persone che comunicano sono presenti contemporaneamente. Mentre i protocolli e le applicazioni più diffuse rendono possibile la comunicazione solo con testi, in altri casi è possibile anche comunicare in forma parlata ed in video, quando si ha a disposizione una banda sufficiente.

IRC (IRCP, Internet Relay Chat Protocol RFC1459)
IRC significa "Internet Relay Chat" ed è un servizio di trasporto (relay) via Internet di messaggi di tipo testuale in chat da molti a molti. Il protocollo IRCP è contenuto nella RFC1459.
Gli utenti di IRC si incontrano in "stanze" dette "canali", nelle quali si dovrebbe parlare dello stesso argomento.
L'argomento di discussione può essere pubblicizzato nel titolo del canale, che può essere visto da chiunque si colleghi ad uno qualunque dei server che fanno parte di una specifica rete IRC.
Fra utenti che si incontrano nello stesso canale è possibile stabilire connessioni da uno a uno, per chiacchiere "riservate".
Per chiacchierare in IRC è indispensabile avere un programma client che deve rispettare il protocollo IRCP, scritto in Finlandia nel 1988 ed in seguito divenuto protocollo Internet. Il client si può collegare ad una delle tante reti IRC esistenti. Una rete IRC è un insieme di server che comunicano fra di loro usando l'Internet e scambiandosi informazioni che riguardano: gli utenti che sono collegati, i canali che sono presenti ed i messaggi che ciascuno degli utenti vuole scrivere in ciascuno dei canali. Quando arriva un nuovo messaggio destinato ad uno dei canali ai quali siamo collegati, il server IRC lo manderà al client che lo presenterà nella finestra relativa a quel canale, con l'indicazione del soprannome dell'utente che lo ha spedito.
Un canale IRC si crea quando il primo utente ci "entra", se non c'è più nessuno il canale sparisce. È possibile creare degli utenti artificiali, che in realtà sono dei programmi residenti sul server, che vengono detti "robot" o semplicemente "bot". Un robot, essendo artificiale, non si stanca mai e non esce mai dal canale, rendendolo disponibile in ogni momento. Esso può avere anche funzioni di "poliziotto" escludendo utenti indesiderati e dando autorizzazioni per il controllo del canale solo agli utenti autorizzati.
In una rete IRC gli utenti devono tutti avere nomi diversi. Dato che in ogni istante ci possono essere anche 15 000 persone contemporaneamente in linea, ci si dovrà scegliere un nome "improbabile", altrimenti il server rifiuterà la connessione. Il nome che si usa non deve essere necessariamente il proprio, né è obbligatorio fornire, quando si fa il collegamento, un indirizzo di posta elettronica esatto.

Chat multimediali e videoconferenze
Esistono protocolli ed applicazioni che permettono uno scambio di informazioni non solo testuali. In alcuni casi i vari utenti vengono "impersonati" da loro alter ego virtuali, detti "avatar" che assumono l'aspetto che vuole l'utente, per esempio quello di un particolare un personaggio a fumetti, come avviene in "Microsoft chat". In questo caso, oltre che comunicare in modo testuale, l'utente potrà simulare meglio la comunicazione fra esseri umani facendo assumere al suo personaggio le espressioni facciali volute.
Il passo successivo è quello di scambiarsi file mentre si chiacchiera, in modo da migliorare la qualità dell'interazione. In diversi programmi quali Pow-Wow o MS NetMeeting, nel quale è anche possibile disegnare e scrivere sulla stessa "lavagna elettronica", condivisa da tutti i partecipanti al chat.

Altre applicazioni permettono di parlare con la propria voce, e qui si sconfina nella telefonia Internet, o addirittura mostrando la propria immagine, e qui si parla di videoconferenza. Naturalmente nei collegamenti lenti o quando la rete è congestionata la qualità del suono e dell'immagine è molto scadente, nonostante gli ottimi algoritmi di compressione utilizzati.
NetMeeting rende possibile la videoconferenza, ma con un solo collegamento in ricezione video per ogni utente, mentre "CU see me" (see you see me) da' un collegamento video molti a molti.

Strumenti per la rilevazione della presenza in rete (instant messager)
Finger (port 79)

Protocolli proprietari
AOL – ICQ, Microsoft, CuSeeMee, Jabber

Workflow

Controllo delle versioni dei file (file versioning)
CVS, SourceSafe

1.1.5 HTTP (HyperText Transfer Protocol, port 80, RFC2068 no STD)

Il protocollo HTTP realizza un semplice meccanismo per lo scambio di "risorse" (file) fra applicazioni, ottimizzato per la trasmissione di ipertesti scritti nel linguaggio HTML (**H**yper**T**ext **M**arkup **L**anguage). Esso ha bisogno di una comunicazione affidabile e con collegamento "end to end" fra un client ed un server. Come già sappiamo, questo tipo di comunicazione viene messa a disposizione da TCP.
In una comunicazione che usa HTTP si possono distinguere un "server" (server HTTP o "**Web server**"), che distribuisce le informazioni, ed un "client", che le legge. Il client HTTP viene detto "browser"; esso scarica ipertesti HTML da Internet via HTTP e li visualizza. L'operazione di visualizzazione, che da un file HTML produce una pagina Web, viene detta "rendering" della pagina.
Il browser chiede al server Web le pagine HTML utilizzando il protocollo HTTP, mentre HTML è il linguaggio che descrive le informazioni ipermediali visualizzabili dal browser.

Il trasferimento di ogni file richiede l'esecuzione di una sequenza composta delle seguenti fasi:
1. Client: richiede l'inizio di un collegamento TCP con il server (default port del server: 80)
2. Server: accetta la richiesta di collegamento (se è il caso!). Viene stabilito un collegamento virtuale TCP fra il client ed il server.
3. Client: richiede al server, sul canale (stream) appena creato, il documento che vuole
4. Server: invio del documento da server a client, sullo **stesso** stream
5. Chiusura del socket TCP da parte del server
6. I server HTTP dalla versione 1.1 possono, su richiesta del client, evitare di chiudere il collegamento alla fine della spedizione del file. Ciò per questioni di efficienza, dato che possono usare il collegamento lasciato aperto per la prossima richiesta di file da parte del client; perché ciò sia possibile il client deve sapere che il server è in grado si funzionare in questo modo (lo ricava dall'header spedito dal server).

In HTTP tutto ciò che viene scambiato fra client e server, anche i comandi al server e le sue risposte, sono composti di una parte iniziale obbligatoria ("header"), seguito da una parte dati, che può anche mancare.

Header HTTP del client
L'header HTTP è costituito da alcune linee contenenti comandi ed altre eventuali informazioni ed è concluso da una linea vuota.

Formato dell'header HTTP del client
< *campo "metodo"* >
< *campi "richiesta"* >
< *Linea vuota* >

Ogni campo è concluso da <CRLF>
Il primo campo dell'header HTTP è il "method field" (campo "metodo") ed indica il comando che il client sta spedendo al server, detto "metodo" in HTTP.
Esso ha la forma:
< *campo "metodo"* > := < *"metodo"* HTTP > < *identificatore* > < *versione di* HTTP > < CRLF >

<"metodo" HTTP> è una stringa che indica il comando HTTP che il client rivolge al server. Alcuni del metodi HTTP sono elencati poco più avanti
<identificatore> è una stringa che identifica la risorsa su cui il server deve applicare il comando (tipicamente si tratta del percorso e del nome di un file)

<versione di HTTP> è la versione più aggiornata di HTTP con cui il client è in grado di funzionare.
<CRLF> Il campo metodo viene concluso da un "a capo" ASCII, cioè dai caratteri "Line Feed" (LF, codice ASCII 10), e "Carriage Return ("CR", codice ASCII 13).

Dopo il campo metodo che è una sola "linea" dello stream ASCII che va nella direzione dal client al server, seguono, in nuove righe, uno o più "campi richiesta", tramite i quali il browser fornisce informazioni addizionali al server.
P.es. nell'header il browser può indicare quali sono i tipi MIME che esso è in grado di accettare. Queste informazioni verranno incluse in un "campo accept", come nell'esempio seguente.
Il browser potrà inoltre includere altre informazioni, che potranno essere utili al server per meglio "servirlo" (vedi esempio).

Esempio di comando HTTP GET, spedito al server da parte del client, come header "solitario", senza parte di dati.
Ciò che è scritto qui sotto sono i "veri" caratteri che passano sullo stream dati al port 80 del server, dopo la connessione, nella direzione da client a server, corrispondono alla richiesta al server del file /index.html:

```
GET /index.html HTTP/1.0
^^^^ campo metodo richiesta al server, con metodo "GET", del file di nome index.html,
^^^^ dalla sua cartella "radice". Il client dichiara anche con quale versione di HTTP
^^^^ è in grado di funzionare ^^^^
User-Agent: Mozilla/1.N (Windows; I; 32bit)
^^^^ indicazione del browser e della versione. Conoscendo questa informazione, il server
^^^^ potrà restituire file HTML diversi in base al browser dichiarato
^^^^ I file forniti saranno così meglio adatti al suo funzionamento di tale versione del browser
Accept: image/gif
^^^^ il browser dichiara che è in grado di visualizzare immagini di tipo GIF ^^^^

^^^^ LINEA VUOTA che attesta la conclusione dell'header
^^^^ Qui seguirebbe il vero e proprio messaggio, se fosse un altro comando
^^^^ ma il messaggio, in un comando GET, non c'è!
```

In questo caso all'header non fa seguito null'altro; il collegamento potrebbe anche essere chiuso.
Non appena il server riceve la linea vuota, esso esegue il comando GET.

I comandi HTTP ("metodi" da client a server)
GET <path e file di un URI> [<estensioni dell'URI>]
Richiede al server HTTP la spedizione di un file, dopo la path ed il nome del file possono essere presenti altri argomenti, che forniscono parametri ad un programma che gira sul server (<estensioni dell'URI>). Su queste estensioni, vedi oltre.
Se non è presente alcun nome di file, il server andrà nella cartella indicata e spedirà la pagina di default per quella cartella, che potrebbe essere, per esempio, index.html (tipica per server Unix), o default.htm (tipica per server Windows).
Se la cartella indicata è / (comando GET /) il server cercherà nella cartella "radice"[2] del sito (home directory) e fornirà la "home page" del server.
Configurando il server si può cambiare a piacimento: il nome della cartella "radice" ed il nome del documento di default (index.html, default.htm, default.asp, index.php, ..) , in ciascuna delle directory (cartelle) del server HTTP.

POST <path e nome di un file>
Spedisce dati dal client al server. I dati hanno il nome di file specificato da <path e nome di un file>.
I dati seguono dopo l'header HTTP, nella parte che anche il server usa per spedire i suoi file. Questo comando viene usato dai browser anche per spedire i dati che l'utente ha scritto nelle pagine interattive (FORM, vedi oltre).

HEAD <path e file di un URI>
Con Head il client chiede di spedire il solo header HTTP del file indicato, che fornisce informazioni sul file stesso. NON verrà spedito il file. Con questo comando potrà sapere, p.es. quanto è lungo il file e qual è la data della sua ultima modifica, oltre ad altre informazioni.

LINK

UNLINK

PUT

TEXTSEARCH

2 Di solito è diversa dalla cartella radice del computer che ospita il server.

La risposta del server, header e dati HTTP del server

In risposta ad un comando il server passa al client dati che hanno la stessa struttura già vista: un header ed una parte dati, opzionale. Se il client aveva richiesto un file con una GET, nella parte dati sarà incluso quel file, in un formato MIME, analogo a quello utilizzato per gli allegati di posta elettronica.

Header HTTP del server

L'header HTTP del server inizia con una linea di stato che dà informazioni sul server e sulla richiesta presentata dal server, poi seguono campi di informazione varia, da consegnare al browser perché agisca di conseguenza.

Formato dell'header HTTP del server:
<Versione HTTP> <Codice di stato> <Spiegazione del codice di stato>
<campi di informazione>
<Linea vuota>

<campi di informazione> sono una o più righe che trasmettono informazioni sul server e sul suo stato, ad uso del browser.
Dopo la linea vuota che conclude l'header può esserci il contenuto di un file (p.es., se il messaggio è in risposta ad un comando GET).

Esempio di risposta HTTP alla GET precedente, spedita al client da parte del server, e che include la "home page" HTML.
Ciò che è scritto qui sotto sono i caratteri che passano sullo stream dati al port 80, dopo la connessione, in direzione da server a client:

```
HTTP/1.0 200 Document follows
^^^^ la prima riga dichiara la versione ed indica che il file richiesto ^^^^
^^^^ esisteva ed è incluso in questo messaggio (codice 20) ^^^^
Date: Tue, May 12 2003
^^^^ data e ora della spedizione di questo messaggio^^^^
^^^^ l'ora indicata è nel fuso di Greenwhich (GMT = Greenwhich Mean Time) ^^^^
^^^^ file diversi in base al browser dichiarato, meglio adatti al suo funzionamento ^^^^
Server: NCSA/1.4.1
^^^^ il server si dichiara ^^^^
Content-type: text/html
^^^^ il server dichiara di che tipo MIME è il file incluso in questo messaggio ^^^^
Last modified: Fri, May 9 2003 16:23:12 GMT <CRLF>
^^^^ il server dichiara la data di ultima modifica del file richiesto ^^^^
Content-lenght: 198
^^^^ il server dichiara la lunghezza in byte del messaggio che segue ^^^^

^^^^ LINEA VUOTA che attesta la conclusione dell'header ^^^^
<html>
^^^^ prima linea del corpo del messaggio: è la prima linea del file HTML ^^^^
^^^^ segue il testo di tutto il file richiesto ^^^^
<head>
    <title> Home page del sito </title>
</head>

<body>
    <h1>Home page di prova del sito </h1>
    <br>    <br>
    <A HREF="index.php">Vai alla "vera" home page</A>
</body>

</html>
^^^^ ultima linea del corpo del messaggio; qui il collegamento viene chiuso ^^^^
```

Server HTTP: i programmi

Il mercato dei server HTTP per uso generale è al giorno d'oggi suddiviso fra Apache, software libero con licenza GPL, che può eseguire sia su Linux che su Windows, e Internet Information Services (IIS) di Microsoft, che è il software "nativo" di Windows, presente nelle versioni "server" e "professional", e che può girare solo su tale sistema.
Esistono inoltre server speciali per applicazioni embedded, dato che molte apparecchiature di rete vengono configurate attraverso il browser, montando un server HTTP.
Esistono server HTTP su: router, switch ed altre apparecchiature di rete, telecamere per la sicurezza e WebCam, server per la configurazione remota dei computer (p.es. il server Webmin, che, usando HTTPS, permette di configurare Linux da remoto).

HTTPS

La RFC2818 "HTTP Over TLS", Maggio 2000, specifica una versione sicura di HTTP che critta tutti i dati che spedisce (HTTPS, **HTTP S**ecure).
TLS (o SSL, Secure Socket Layer) è una versione dell'interfaccia software a socket che aggiunge funzioni di crittazione. Con SSL tutti i dati che vengono spediti nei segmenti TCP vengono crittati alla fonte e viaggiano crittati su tutta la rete, venendo decrittati solo a destinazione.
TLS fa uso della crittografia a chiave pubblica, con algoritmo RSA, e dei **certificati** digitali.
Il server ed il client HTTPS si scambiano due certificati digitali la prima volta che si scambiano dati. In questo modo sono in grado di accertarsi che i dati provengano proprio dal server voluto e che non siano stati alterati lungo il loro percorso.
Per il collegamento con server "sconosciuti" i certificati possono essere riconosciuti da una delle Aziende che svolgono la funzione di "Autorità di certificazione". Questa Azienda memorizza i certificati elettronici dei suoi clienti sui suoi server, rendendoli disponibili a tutti su Internet. Dato che l'autorità di certificazione ha identificato con precisione i propri clienti, si genera uno schema efficace per l'assicurazione dell'identità dei soggetti che si "incontrano" in Internet.
Quando un certificato giunge alla sua scadenza naturale o viene revocato dal suo proprietario, l'Autorità di Certificazione lo pubblica sui propri server, in modo che tutti possano controllare in tempo reale se il certificato non è più valido.

1.2 HTML (HyperText Markup Language, versione 2 RFC1866)

Un file HTML è un normale file di tipo testo, costituito da caratteri in codice ASCII o Unicode. Un testo HTML deve poter identificare collegamenti "ipertestuali" ad altre "risorse". Un collegamento "ipertestuale" (**iperlink**) è un riferimento ad un altro documento. I file HTML vengono interpretati e visualizzati da un particolare programma, detto "browser". Il browser identifica i collegamenti in modo visibile, così che i documenti relativi possano essere caricati e visualizzati con un semplice click del mouse in un'area dello schermo. Si possono visualizzare file che contengono informazioni espresse con "mezzi di comunicazione" diversi (multi - media). Infatti un iperlink può essere un file di testo, un altro documento HTML, un'immagine a due o tre dimensioni, un suono, un video, un programma in sorgente o in forma binaria, in generale un qualsiasi tipo di file che possa essere interpretato e "visualizzato" dal browser. L'operazione di trasformazione dei file nel formato multimediale fruibile all'utente viene detta "**rendering**" della pagina HTML.

Come accennato, ogni riferimento ipertestuale di HTML è un file. Nel gergo HTML esso si chiama "risorsa". Per avere acceso ad una risorsa HTML è necessario disporre di un nome univoco che la identifichi nella rete. Questo nome viene detto URI o, meglio, URI (vedi in seguito).
Il browser deve essere in grado di capire il file HTML, chiedere al server, p.es. con il comando GET, tutte le risorse incluse nel file HTML e "renderizzarle" ("visualizzarle" se si tratta di immagini, "suonarle" se di tratta di suoni ..).
Per capire il tipo del file da renderizzare il browser usa l'header HTTP del file, spedito dal server, ove trova indicazione del tipo MIME. Noto il tipo MIME, il browser fa partire il programma giusto per la renderizzazione.

Deve inoltre accettare tutte le richieste dell'utente e provvedere a che siano esaudite. L'utente può richiedere di visualizzare iperlink, fare input di dati da spedire al server. In molti casi deve anche eseguire programmi che vengono inclusi nei file HTML. Il codice di questi programmi deve essere incluso in file, indicati come riferimento dal file HTML. In questo caso il browser deve essere in grado di compilare od interpretare le istruzioni del linguaggio di programmazione che vengono collegate attraverso quel file.
La versione 1.0 di HTTP è in RFC1945.
HTML è anche interattivo (form e CGI, vedi oltre).
HTML ha estensioni multimediali sofisticate (DHTML, **D**ynamic **HTML**)

URI (già URL)

URI sta per "Uniform Resource Indicator" (RFC2396). Inizialmente era noto come URL, "Universal Resource Locator", (RFC1738) e, come dice il nome, è un "indirizzo" per trovare una qualsiasi "risorsa" in un qualsiasi computer in qualsiasi rete. Per HTTP una "risorsa" è, semplicemente, un file. Gli oggetti cui si può fare riferimento in HTTP sono quindi solo dei file.

Gli URI, nati in HTTP come URL, sono usati oggi anche in molti altri protocolli, anche non "internet". Un URI è una semplice stringa che ha il formato:
URI:= < SchemaURI > **://** < Host > [**:** < NumeroDellaPortaTCPsullHost >] [**//** < PathNeiDirectory >][**//** < NomeDelFile >] [**?** < estensione della path o parametri CGI >]

Il fatto che ci sia uno <*SchemaURI*> ("URI scheme") ci fa capire come un URI possa essere utilizzato con protocolli diversi, non solo con HTTP. Infatti i valori più tipici di <*SchemaURI*> sono:

< *SchemaURI* > := **http** | **ftp** | **nntp** | **mailto** | **nntp** | **telnet** | **file**

cioè molti dei più importanti protocolli di livello "applicazione" della suite Internet ("mailto" sta per SMTP o per POP3; di solito quando lo si usa nel browser esso lancia il mailer, con indirizzo di destinazione quello indicato in mailto).
Perché la risorsa indicata dall'URI sia accessibile il computer che la contiene deve disporre di un software che realizza il protocollo indicato, cioè deve essere un server per quel particolare protocollo.

La IANA mantiene una lista "Registry of URI Schemes", che contiene un elenco di una sessantina di schemi. Come indica lo schema "file" si può indicare anche qualcosa di diverso da un protocollo. Infatti se si usa "file", il browser punta all'hard disk di un computer, se <host> è indicato può essere un computer nella rete locale, se non è indicato è il computer locale (localhost). In entrambi i casi <PathNeiDirectory>/<NomeDelFile> indicano dove reperire il file sul computer indicato.
Quando lo schema indica un protocollo, e se non è indicato il <NumeroDellaPortaTCPsullHost>, il browser contatta il server al port indicato nella tabella dei "well known port numbers".

In HTTP gli URI possono "trasmettere" al server dati immessi dall'utente nella pagina HTML del browser.
Questi dati vengono acquisiti dal server e passati ai "programmi CGI". Se il browser passa i suoi dati utilizzando il metodo GET di HTTP essi vengono aggiunti in fondo all'URI (<parametri CGI>).

L'indicazione dell'host può essere data come indirizzo IP o come nome (nome di dominio DNS o nome del computer in una rete locale):
<Host> := <NomeDelDominio> | <IndirizzoIP>

Formati particolari di URI, in casi nei quali il protocollo sia HTTP:
Se il nome del file non è indicato (*<PathNeiDirectory> e <NomeDelFile>* sono opzionali) il server HTTP dell'host raggiunto fornirà automaticamente il documento di default che corrisponde alla path indicata (es. "index.html", o "default.htm").

<NumeroDellaPortaTCPsullHost> se il protocollo è HTTP è per default 80 (se non indicato il browser cercherà il server HTTP al port 80).

Evoluzioni: CSS, XML

WWW Worldwide Web

W3C
Il "WWW consortium" (W3C) è l'organizzazione che sovrintende ai linguaggi ed ai protocolli per il Web, quali in particolare HTML.

WebDAV

1.2.1 Programmi "gateway"

Il server HTTP (Web server), su richiesta del client, può far eseguire particolari programmi, che di solito hanno il compito di elaborare i dati trasmessi dal client, produrre "al volo" una pagina HTML, e di spedirla in ritorno al client.
Questi programmi sono detti "Programmi Gateway", perché fanno da "ponte" fra il client ed il server, o programmi CGI. Il nome deriva dal fatto che questi programmi rispettano la "Common Gateway Interface", o "CGI". CGI è una specifica che indica come si devono comportare i programmi per poter interagire con il server ed il client HTTP.
Con i programmi CGI si può dunque realizzare **pagine Web dinamiche**, che cambiano sotto controllo del programma, in base agli input dell'utente.
C'è dunque un programma dal lato del server ("server side") che elabora i dati dell'utente e produce pagine HTML dinamiche.
I programmi CGI possono essere scritti in molti linguaggi, alcuni interpretati, altri compilati; per esempio: PERL, Unix shell script, TCL, C/C++.
Scrivere programmi che rispettino le specifiche CGI è piuttosto difficile e noioso, per cui sono state scritte librerie che agevolano la programmazione CGI con praticamente tutti i linguaggi di programmazione "general purpose". Inoltre sono stati sviluppati linguaggi ed "ambienti" di programmazione fatti apposta per comunicare con il Web server e realizzare programmi CGI in modo agevole. I più importanti di questi strumenti, il cui uso copre la stragrande maggioranza della programmazione "lato server" sono: il linguaggio PHP e l'"ambiente" di programmazione ASP (o ASP.NET), Java Server Pages (JSP).

Esempi:
Il seguente programma è uno "shell script" CGI per Linux, cioè una sorta di file .BAT per Unix, che esegue i comandi che sono scritti nel file come se fossero digitati in successione alla tastiera.

```
#!/bin/sh        # dichiara che questo script usa la shell sh
# script CGI di prova; inizio dell'header HTTP
echo "Content-type: text/html"
echo             # linea vuota, fine dell'header!
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE> Script CGI di prova, titolo </TITLE>"
echo "</HEAD>"
echo "<BODY>"
```

```
echo "<H1> Script CGI di prova, testo nel BODY, Heading 1 </H1><br>"
echo "L'utente che fa girare questo programma è <B>"
whoami           # comando di sistema che scrive nello standard output
                 # lo username viene visualizzato in grassetto (è preceduto da <B>
                 # e seguito da </B>:
echo "</B> <br><br>"
echo "Ora mostrerò i processi che fanno girare questo server Web: <br>"
ps aux | grep httpd  # un altro comando di sistema.  L'output di questo comando
                     # viene formattato "male" dal browser, perché per andare a
                     # capo usa solo il carattere "Carriage Return" e non <br>
echo "</BODY>"
echo "</HTML>"
```

Questo "script" mostra come il file HTML venga costruito "a mano", con istruzioni echo; anche l'header HTTP viene generato in questo modo. Oltre alle echo sono presenti nello script anche due comandi di sistema: whoami e ps aux | grep httpd, che producono il loro "normale" output, cioè il nome dell'user Unix che fa eseguire il server, poi l'indicazione dei processi che costituiscono il server (httpd). L'uscita dei due comandi, invece che essere spedita al terminale, viene spedita al client.

Il programma (script) appena illustrato manda lo standard output, tale e quale, invece che al terminale, al client HTTP. Per poter essere usato dal server lo script precedente deve essere "richiesto" dall'utente, p.es. cliccando su un bottone generato da un comando FORM, incluso in un file HTML.
Se lo script precedente viene incluso in un file di nome "processi.sh", la seguente pagina HTML permette all'utente di farlo eseguire sul server.

```
<HTML>
<BODY>
<FORM ACTION="/cgi-bin/processi.sh">
    <INPUT TYPE="submit" value="Vedi utente e processo">
</FORM>
</BODY>
</HTML>
```

Passaggio di parametri ai programmi gateway
Vediamo ora come si può passare parametri ai programmi gateway, con i metodi GET e POST e l'istruzione HTML FORM.

Il seguente file si chiama textBoxForm.html e contiene la definizione di un form che viene spedito con il metodo GET, aggiungendo i parametri nell'URI.

```
<HTML>
<HEAD>
<TITLE> File HTML con un FORM contenente text box, uso del metodo GET </TITLE>
</HEAD>

<BODY>
<FORM ACTION="/cgi-bin/input_superiori.sh" METHOD="GET">

<H1> MODULO RISERVATO AI SUPERIORI </H1>
Modulo per il megapremio di fine anno <br><br>
Nome    <INPUT TYPE="text" NAME="Nome"  VALUE="Vittorio Emanuele" > <br>
Cognome <INPUT TYPE="text" NAME="Cognome"  VALUE="MAZZANTI VIENDALMARE" > <br>
Titolo  <INPUT TYPE="text" NAME="Titolo"  VALUE="Conte Duca"> <br><br>
<INPUT TYPE="submit" value="Spedischi i dati">
</FORM>
</BODY>
</HTML>
```

Questo è il file "input superiori.sh". Esso è uno shell script Unix che non fa altro che "mostrare in HTML" il valore della variabile d'ambiente QUERY_STRING. Questa variabile contiene i parametri passati dal metodo GET ai programmi gateway:

```
#!/bin/sh
# script CGI di risposta all'input proveniente da textBoxForm.html
echo "Content-type: text/html"
echo             # linea vuota, fine dell'header!
echo "Eccellentissimo $QUERY_STRING <br><br>"
# ^ visualizza la QUERY_STRING così come viene ricevuta dal programma gateway
echo "Siamo felici di comunicarCi che il suo premio produzione è di Euro 1 000 000"
```

Se l'utente non cambia i dati di default del precedente file textBoxForm.html il valore di QUERY_STRING, visualizzato dallo script è:

```
&Nome=Vittorio+Emanuele&Cognome=MAZZANTI+VIENDALMARE&Titolo=Conte+Duca
```

Vediamo ora un HTML client che usa delle liste di scelta a scomparsa ("combo box"), che indicano alcuni valori fra i quali l'utente deve scegliere; il relativo modulo HTML fa uso del metodo POST:

```
<HTML>
<HEAD>
<TITLE> File HTML con un FORM contenente combo box, uso del metodo POST </TITLE>
</HEAD>
<BODY>
<FORM ACTION="/cgi-bin/input_inferiori.sh" METHOD="POST">
<H1> MODULO RISERVATO AGLI INFERIORI </H1>
Caro inferiore <br>
Data la tua scandalosa inefficienza devi scegliere la tua punizione: <br>
<SELECT NAME="selezione">
<OPTION SELECTED VALUE=1> Un Sabato in ginocchio sui ceci </OPTION>
<OPTION VALUE=2> Cancellazione delle ferie per due anni </OPTION>
<OPTION VALUE=3> Crocifisso in sala mensa </OPTION>
</SELECT>
<br><br>
<INPUT TYPE="submit" value="Spedisci i dati">
</FORM>
</BODY>
</HTML>
```

Se supponiamo che il file input_inferiori.sh sia identico al input_superiori.sh, esso non visualizzerà i dati dell'utente, perché vengono passati attraverso il suo standard input e non attraverso la variabile QUERY_STRING, che perciò è vuota.
Se invece input_inferiori.sh è il seguente:

```
#!/bin/sh
# script CGI di risposta all'input proveniente da comboBoxFrom.html
echo "Content-type: text/html"
echo             # linea vuota, fine dell'header!
echo "La punizione  che hai scelto è: <br> "
cat
# cat spedisce sullo standard output lo standard input => il client vede
# ciò che ha spedito con il metodo POST attraverso lo standard input.
```

Con uno script come il precedente il risultato sul browser potrebbe essere:

```
La punizione che hai scelto è:
&Selection=3
```

Se Fantozzi ha scelto di essere "Crocifisso in sala mensa" (value=3).

1.2.2 Programmi "server side"

PHP, ASP

1.2.3 Whois(port 43)

1.2.4 NTP (Network Time Protocol, RFC1119 STD 12)

Alcuni protocolli della suite Internet hanno bisogno di effettuare operazioni nelle quali si deve avere un riferimento temporale preciso. Ciò può avvenire, per esempio, nel caso in cui i router debbano valutare le prestazioni dei collegamenti, per procedere ad un instradamento ottimizzato. Ogni router può misurare il tempo localmente, ma gli orologi interni, nel corso del tempo, possono perdere il sincronismo. Per rimettersi a punto i router più moderni sono in grado di usare il protocollo **NTP** (**N**etwork **T**ime **P**rotocol, RFC1305, giunto alla versione 3).
In NTP alcuni router o host "speciali" (primary time servers) sono sincronizzati ai campioni nazionali di tempo e lo misurano con buona o grande accuratezza (p.es. esiste un server primario NTP all'IEN Galileo Ferraris di Torino, sincronizzato direttamente al campione di tempo dell'Italia (il campione nazionale permette un'incertezza relativa di $1 * 10^{-13}$)). I time server primari sono di solito collegati ai gateway delle più importanti dorsali di Internet. Altri router o host (secondary time servers) utilizzano uno o più time server primari come riferimento per sincronizzarsi con essi. A sua volta, un host qualsiasi che è in grado di usare NTP, si può sincronizzare al clock "della rete" utilizzando un gruppo di time server secondari. Il protocollo è basato su IP e UDP ed è progettato per mantenere la sua accuratezza anche su Internet, con percorsi multipli in molti gateway e ritardi molto variabili. Il protocollo permette di determinare una stima dell'errore ("offset") del clock locale, che può essere aggiustato automaticamente in conseguenza dell'errore misurato. I diversi time server sono "mutuamente sospettosi", il protocollo prevede algoritmi per individuare i clock buoni e quelli non affidabili nel gruppo di quelli utilizzati per la sincronizzazione. L'accuratezza raggiungibile è comunque dell'ordine dei 100 - 200 ms e dipende fortemente dalla stabilità del clock locale (quello del dispositivo che si sta sincronizzando).

In linea di principio ed usando le migliori apparecchiature esistenti (orologi "atomici" al cesio), l'accuratezza è in grado di arrivare fino al ns. Esiste una versione semplificata di NTP di nome **SNTP** (**S**imple **N**etwork **T**ime **P**rotocol, RFC2030), che usa un solo time server e viene usato per le normali stazioni di rete. Un programma gratuito che fa la sincronizzazione con SNTP è "Dimension Four".

1.2.5 SNMP (RFC1157 STD 15)

Simple **N**etwork **M**anagement **P**rotocol è un protocollo che permette di gestire da remoto computer, switch, router e molti altri tipi di apparecchiature collegate in rete.

SNMPv2 RFC1441

1.2.6 Applicazioni diagnostiche

Ping
Ping verifica il funzionamento di un collegamento IP, spedendo all'indirizzo desiderato un datagramma ICMP che contiene il comando ECHO. Quando l'host desiderato riceve il comando ECHO non fa altro che rispedirlo al mittente, uguale a come era arrivato. Dunque il ritorno di un ping significa che la rete, a livello IP, funziona.
Non è detto che tutti gli host rispondano comunque al Ping. Infatti, per questioni di sicurezza, alcuni amministratori disabilitano questa funzione.

TraceRoute
Traceroute è un programma, presente sia nei sistemi Unix, con il nome "traceroute", che nei sistemi Windows, con il nome di "tracert" (con 8 caratteri).
Questo programma tenta di determinare la strada che viene percorsa dai datagrammi per giungere ad una qualunque destinazione a partire da un qualunque host sorgente. Esso si basa sull'assunzione che la strada percorsa dai pacchetti, pur essendo dinamica, non cambi nel periodo di tempo richiesto per la determinazione della "traccia".
Il programma manda all'indirizzo IP di destinazione datagrammi IP ICMP di tipo "echo" (lo stesso usato dal programma "ping"), con numero "Time To Live" (TTL) crescente, a partire da 1.
Il primo di questi datagrammi verrà scartato dal primo dei router del percorso, dato che TTL =1 diventa zero proprio nel primo router. In conseguenza del fatto, questo router dovrebbe spedire al mittente un messaggio ICMP ("Time Exceeded") che l'avverte che il suo datagramma è stato perso. Il mittente è dunque in grado di sapere l'indirizzo IP del primo router, leggendo l'indirizzo del mittente della comunicazione d'errore di ritorno.
Poi traceroute spedisce una richiesta echo con TTL = 2, e sarà il secondo router a rispondere, e così via, fino a che il "ping" non otterrà risposta dall'host cui era destinato (se l'host risponde ai ping!).
Ad ogni passo di questo processo traceroute può anche fare un reverse lookup del DNS in modo da indicare, insieme all'indirizzo IP del router, anche un suo nome simbolico.
C'è da dire però che al giorno d'oggi moltissimi router scartano i datagrammi IP con TTL = 0 senza comunicare nulla al mittente del datagramma, ciò perché "tempeste" di questi ICMP sono state usati da malintenzionati per bloccare importanti computer sull'Internet (attacco di tipo "Denial of Service" DOS). Questi router saranno dunque invisibili a traceroute (Si otterrà per il loro "hop" la risposta: "No response from this host").

1.3 Alcuni dispositivi e software speciali nel mondo Internet

Router NAT (RFC 1631)
Un router NAT fa la "Network Address Translation". Esso modifica i datagrammi IP che lo attraversano, sostituendo indirizzi pubblici che provengono da Internet con indirizzi privati che provengono dalla LAN.
Funzionamento di un router NAT "puro":

- semplice associazione temporanea fra diversi indirizzi privati e diversi indirizzi pubblici. In un certo istante un indirizzo pubblico corrisponde ad un solo indirizzo privato. In questo caso il lavoro un un router NAT è solamente quello di associare temporaneamente i due indirizzi e sostituirli. Nei datagrammi che entrano sostituisce l'indirizzo pubblico con quello privato ad esso temporaneamente associato. Nei datagrammi che escono sostituisce l'indirizzo privato con il "suo" indirizzo pubblico.

Per dar luogo a comportamenti più flessibili e sofisticati, un router NAT deve far uso anche dei numeri di port TCP od UDP, che gli permettono di associare un singolo indirizzo IP pubblico a molti indirizzi IP privati. In questo caso diventa un router NAT/PAT (detto anche solo "PAT router").
Disegno

Router PAT
Un router che fa la "traduzione" sia degli indirizzi IP, sia dei numeri di port TCP/UDP viene detto NAT/PAT (**P**ort-Address Translation) router.

Connessioni che originano dall'interno (LAN)
Le connessioni che originano dalla rete locale e sono rivolte all'Internet usano quasi sempre un numero di port remoto specifico (tipicamente il "well known port number" del protocollo). Il port locale è invece un numero qualsiasi, maggio-

re di 1024 e minore di 64k). Dunque il router PAT può sostituire liberamente il numero di port locale, mettendone uno diverso nei datagrammi che spedirà in Internet. Quando l'host Internet contattato dalla nostra LAN risponderà, esso indicherà il numero di port "sostituito" dal PAT. Questo numero di port servirà al router PAT per "ricordarsi" qual era l'indirizzo IP originale, della LAN, e per sostituirlo nel datagramma di ritorno, ripristinando anche il numero di port originario.
Questa tecnica viene chiamata "**IP masquerading**".
Il router PAT usa il numero di port locale[3] dell'host che tenta di uscire sui Internet per ricordare da dove veniva la richiesta di collegamento. In questo modo può ridirigere all'indirizzo IP privato tutto ciò che proviene dall'esterno ed è indirizzato al port che ha "tradotto" verso l'esterno.
E' chiaro dunque che un router PAT non è in grado di fare il "masquerading" delle comunicazioni che non facciano uso di TCP e UDP, e dunque che non abbiano il numero di port.
Disegno

Connessioni che originano dall'esterno (Internet)
Il router PAT può essere configurato per trasferire tutti i dati che vengono dall'esterno e sono diretti ad uno specifico port verso un determinato indirizzo e port della rete locale ("**port forwarding**"). In questo modo si può far sì, per esempio, che un server HTTP all'interno della LAN risponda alle richieste provenienti dall'Internet. Infatti basta configurare il router in modo che tutti i tentativi di collegamento provenienti dall'esterno verso il port 80 (HTTP) vengano diretti all'indirizzo IP privato del nostro Web server ed al port sul quale quel server ascolta.
Questa funzione viene anche chiamata "**Virtual Server**", in quanto il PAT router appare all'esterno come se fosse un server per il protocollo "ridirezionato", mentre il server vero è nascosto dal NAT all'interno della rete locale. Un altro nome di questa funzionalità è "**port mapping**".

Disegno

1.3.1 Proxy server

Un "**proxy**" è dispositivo od un programma che si pone in mezzo fra l'utente finale (client) ed un server, effettuando operazioni di controllo e di memorizzazione.
In particolare il proxi:
1. cattura tutte le richieste che il client fa al server
2. "filtra" le richieste in base a criteri configurabili
3. se le richieste sono "lecite" le presenta al server
4. ottiene risposta dal server
5. memorizza la risposta del server in una sua "cache", per ottimizzare l'uso della rete
6. porge al client la risposta ottenuta dal server

Dato il suo funzionamento, è chiaro che il proxi è un dispositivo che lavora al livello dei server, quindi "di livello 7". E' anche chiaro che il suo funzionamento riguarda un certo, specifico, protocollo e che quindi devono esistere proxi per HTTP, per FTP o per tutti gli altri protocolli che si vogliano controllare con questo strumento. E' peraltro possibile che proxi per protocolli diversi siano presentati all'interno di un unico dispositivo o programma.
Tutti i computer della rete locale si rivolgono al proxy per tutto il loro traffico Internet relativo ai protocolli che il proxy stesso gestisce.

Vantaggi in termini di prestazioni
Il proxi raccoglie tutte le richieste che gli arrivano dalle stazioni di una rete locale e memorizza localmente, all'interno della sua "cache", le risposte dei server esterni.
Se da parte della rete locale arrivano altre richieste per dati già compresi nella cache del proxy, esso:
1. chiede al server esterno se il file richiesto è cambiato rispetto a quello memorizzato localmente
2. se non è cambiato, fornisce il file prelevandolo dalla cache, quindi molto più velocemente rispetto all'accesso ad Internet
3. se è cambiato, lo chiede nuovamente al server esterno, lo memorizza nella cache, e fornisce al richiedente la versione aggiornata

Di solito i dati della cache sono cancellati quando diventano più "vecchi" di una certa soglia temporale, che l'amministratore del sistema può stabilire.
Se in una rete è probabile che due persone diverse cerchino di accedere contemporaneamente allo stesso sito Web le prestazioni nell'accesso ad Internet possono aumentare in modo molto consistente.

Vantaggi in termini di "controllo"
Una funzione importante del proxy è quella di filtro, che impedisce alle richieste verso server che appartengano ad una "lista nera" di uscire e/o entrare dalla rete locale.
Il protocollo HTTP prevede un meccanismo standard per le comunicazioni con il proxy server, mentre ciò non succede per gli altri protocolli di Internet (POP3, FTP, NNTP). Perciò il modo con cui l'accesso degli utenti viene verificato dal proxy dipende dal proxy e non è standard.
I proxi possono avere anche funzioni relative alla sicurezza, che sono caratteristiche anche dei firewall (vedi oltre).

3 TCP o UDP

1.3.2 Firewall

Un sistema che esamini tutti i dati che passano fra due interfacce di rete e li blocchi se non rispondono a determinate regole, viene detto **firewall** ("muro antifuoco"!).
Un firewall potrebbe analizzare e filtrare solo datagrammi IP ed allora sarebbe un "puro" router, od anche segmenti TCP, e verrebbe chiamato "**packet filter**". Se invece fosse a conoscenza del funzionamento dei protocolli di livello applicazione ed espletasse le sue funzioni a quel livello, allora sarebbe un "**proxi firewall**". E' chiaro dunque che un "proxi firewall" deve avere un programma diverso per ciascuno dei protocolli che si vogliono filtrare, mentre le funzioni di filtraggio di basso livello, tipiche dei packet filter, vengono configurate ed applicate a tutti i protocolli di alto livello.
Nel mondo reale esistono firewall di tutti i tipi appena definiti, con funzioni ai vari livelli incluse in prodotti unici.
Per evidenti ragioni di sicurezza, un firewall è visto dall'Internet con un unico indirizzo IP pubblico, gli indirizzi delle stazioni comprese nella rete locale interna non esistono all'esterno del firewall. Dunque un firewall fa quasi sempre le funzioni di NAT router.
Un firewall, come un router normale, può essere sia un programma su un computer che un dispositivo autonomo ed isola una internet dalla Internet, permettendo solo comunicazioni controllate. Ha scopo di protezione dei dati interni e di controllo dell'accesso ai dati esterni. Tutti i datagrammi IP rivolti all'esterno del firewall e quelli che dall'esterno provengono devono passare attraverso il firewall, che lascia cadere quelli non autorizzati o "maliziosi" (che tendono a fare del danno od a spiare informazioni riservate).
Il firewall può impedire di entrare e/o di uscire ai datagrammi di un certo protocollo o che provengono o sono destinati ad un certo indirizzo.
Può inoltre bloccare il traffico su certi port TCP, espletando funzioni non da router ma da dispositivo di livello 5, ed in alcuni casi può controllare anche i comportamenti delle applicazioni (livello 7+, p.es. Controllo del lancio di altre applicazioni, controllo della modifica delle applicazioni rispetto all'ultima esecuzione ..)

Firewall di tipo packet filter

Un packet filter (filtro dei pacchetti) può bloccare il traffico IP e TCP sia in ingresso che in uscita, filtrando sia indirizzi che protocolli. Un firewall di questo genere è dunque "solo" un router "filtrante", o tuttalpiù un dispositivo di livello 5 (TCP), che elimina i datagrammi che non rispettano le regole. Per esempio potrebbe bloccare tutti gli accessi di un'azienda all'indirizzo IP di playboy.com, oppure impedire il traffico di tutti i datagrammi telnet o anche di lasciar fare FTP con un server interno ad un ristretto numero di host esterni, compreso in una lista.
Le regole di un filtro di pacchetti possono riguardare:

- l'interfaccia LAN sulla quale vengono scambiati i dati
- gli indirizzi IP del mittente o del destinatario
- il port TCP utilizzato
- il protocollo di trasporto o di applicazione utilizzato
- l'ora e/o il giorno

Nei firewall di tipo "packet filter" il filtraggio può essere applicato:

- sui dati in ingresso dall'esterno verso il firewall
- sui dati che passano da un'interfaccia di rete ad un'altra
- sui dati in uscita dal firewall

Spi (Stateful Packet Inspection)

La principale caratteristica dei migliori firewall è il fatto di poter accettare datagrammi provenienti della rete filtrandoli con regole che tengono conto dello stato della connessione. Sarà perciò possibile, per esempio, far entrare le risposte dei server Web esterni a richieste provenienti dall'interno, ed invece bloccare i segmenti TCP sul port 80 che non sono stati richiesti.

Firewall di tipo server (proxi firewall)

Se il proxi filtra i pacchetti in base a regole più o meno complesse, viene detto "firewall".
Un firewall di questo tipo analizza tutti i dati che riguardano il protocollo con algoritmi più complessi di quelli che può applicare un packet filter.
Funzioni possibili per "proxi firewall" sono:

- Bloccare pagine Web in base a parole in esse contenute
- Bloccare l'accesso a determinati URI

DMZ (DeMilitarized Zone)

La zona demilitarizzata (DMZ) è un insieme d'indirizzi IP che non viene trattato dal firewall. Ogni tentativo di collegamento ad uno di quegli indirizzi IP è direttamente passato all'host che ha quell'indirizzo, di solito attraverso un'interfaccia fisica collegata solo alla DMZ.
I computer della zona demilitarizzata sono raggiungibili direttamente dalla Internet, tramite il loro indirizzo, che deve essere pubblico. Se, per esempio, vogliamo installare un server Web direttamente raggiungibile dall'esterno, dovremo accettare connessioni "non richieste" dall'esterno della nostra rete. Il modo più semplice (ma non l'unico) di lasciarlo liberamente accessibile è metterlo in una zona demilitarizzata.

E' chiaro che la zona demilitarizzata è meno sicura del resto della rete, essendo di fatto al di fuori della protezione del firewall. Per questo deve essere accuratamente tenuta separata dal resto della rete, mettendola in una rete LAN diversa, come indirizzi IP ed anche fisicamente, in modo che un attaccante proveniente dall'esterno che riesca a prendere sotto controllo un computer in DMZ, non possa acquisire informazioni sul resto della rete e non possa provare ad attaccare altri computer fuori dalla DMZ.
La zona demilitarizzata può anche servire quando si debbano usare applicazioni che non sono adatte a funzionare dietro un NAT router. Per esempio, applicazioni che usino direttamente IP, senza usare TCP od UDP, per cui non hanno usare il numero di port. E' il caso di alcuni videogiochi o applicazioni di videoconferenza. Tali applicazioni, non usando il numero di port, non danno modo al router PAT di associare un computer della LAN ad un port "esterno".

1.3.3 Client-side programming

Scripting languages

JavaScript

Java applet

ActiveX, VBscript

Plug-in

1.3.4 Intranet, Extranet

1.3.5 Java

Macchina virtuale, compilazione in byte code

Java come linguaggio general purpose

1.3.6 Tecnologie "push"

1.4 Struttura delle WAN

I componenti principali di una WAN sono due

1.4.1 Nodo di rete geografica

È uno dei dispositivi che realizzano la dorsale della rete geografica. E' in grado di instradare e trasmettere le informazioni a grande distanza. In passato si utilizzava molto X .25, un protocollo a commutazione di pacchetto di provenienza ISO, che riprende in modo piuttosto preciso le indicazioni del modello OSI. Gli apparati X .25 sono attualmente sostituiti con altri molto più veloci: "frame relay" o switch ATM. Altri tipi di protocolli e di apparecchiature, ancora più veloci, che si potranno usare nel futuro sono Sonet e SDH. I nodi di rete geografica sono di solito acquistati e gestiti dai gestori della telefonia pubblica. Pur avendo velocità altissime devono far passare un traffico enorme, per cui ad ogni utente di questi servizi viene riservata una banda relativamente piccola, senz'altro minore di quella delle reti locali.

1.4.2 Nodo di accesso alla rete geografica

È un router multiprotocollo che permette di interfacciarsi da un lato a tipi diversi di reti locali e dall'altro alla rete geografica. Siccome il collegamento con la rete geografica è di solito a velocità molto minore di quello delle reti locali, intervengono in questi sistemi tutta una serie di problemi, che vengono trattati con strumenti sofisticati.
Le tecniche che, non indispensabili nelle reti locali, diventano importanti per l'accesso alla rete geografica, sono quelle relative alla gestione del traffico:

- compressione dei dati, per aumentare la velocità di trasmissione delle informazioni senza aumentare la banda richiesta. Per molti tipi di applicazione la compressione è già realizzata in modo nativo, per cui una ulteriore compressione può far poco
- priorità, per servire per primi quei tipi di trasmissione che hanno vincoli di tempo, come audio e video.
- filtraggio, per non propagare i pacchetti che non devono aver accesso a certe parti della rete, per ragioni di sicurezza o di efficienza
- controllo del flusso, per limitare la congestione della rete

Inoltre è importante poter riservare una certa banda alle trasmissioni che la richiedano. Per esempio alcuni frame relay permettono di suddividere la banda disponibile fra i diversi protocolli, in proporzioni ben determinate. Ogni protocollo

avrà almeno tutta la banda ad esso riservata, quando ne avrà bisogno, mentre essa sarà liberata per gli altri protocolli quando non ne avrà bisogno.
Le apparecchiature che si usano nei nodi di accesso fanno uso delle tecnologie telefoniche dalla parte del collegamento geografico: Multiplexer TDM, ISDN, Frame Relay e ATM.
Al di sopra di questo strato fisico il protocollo che viene più utilizzato attualmente è "Multilink PPP" (RFC1717), che permette di aggregare in un unico canale logico le capacità ridotte di molti canali singoli, di solito linee telefoniche analogiche o ISDN. Questo protocollo permette di aumentare la banda disponibile in condizioni di picco, p.es. effettuando una chiamata telefonica e trasmettendo i dati anche sulla nuova linea. Questa funzionalità viene detta "bandwith on demand".
Router di accesso alla rete geografica vengono prodotti sia dai produttori di reti locali, sia da quelli che si sono sempre interessati del mercato delle reti geografiche.