I PROTOCOLLI

Bene, rieccoci qui ancora una volta per discutere tutti insiemi di un'altra delle componenti fondamentali che tengono in piedi la rete (o le reti che dir si voglia): I PROTOCOLLI.
Ma cosa sono all'atto pratico i protocolli? Se ne sente parlare spesso, e oggi come oggi possiamo affermare che fanno parte integrante della vita di chiunque abbia un modem collegato alla rete.
Detto in parole estremamente semplici, i protocolli sono il formato STANDARD con cui i dati viaggiano in rete. In che senso:
quando noi accediamo a una pagina web, quando inviamo/riceviamo una mail, quando chattiamo, quando scarichiamo un file, quando ci shelliamo su qualche server, non facciamo altro che ricevere e inviare dati. Ma attenzione! Che dati? Solo quelli che a noi compaiono sul monitor? Eh no. I dati che effettivamente vediamo quando ci colleghiamo sono solo una parte di quello che effettivamente si muove. Questi dati fanno solo parte di una complessa struttura che si occupa di far dialogare alla stessa maniera uno, due, tre, mille computer in rete. Questa struttura è composta dai protocolli che impongono il formato con cui i dati devono essere trasmessi/ricevuti. Ogni protocollo è associato a un determinato servizio, ogni protocollo ha le sue regole, ogni protocollo usa il suo formato di comunicazione e soprattutto, ogni protocollo impone il funzionamento per lo svolgersi di una determinata funzione: dall'indirizzamento dei dati, al trasporto dei dati; dalla condivisione dei dati al controllo del funzionamento di un PC connesso in rete e così via.
Ma effettivamente come funzionano? Tutti i dati che trasmettiamo/riceviamo, prima di entrare in rete o di entrare nel nostro computer, devono seguire una "via" obbligata, divisa in più punti. Immaginate di partecipare a una marcia campestre, in cui per poter arrivare alla fine dovete per forza fermarvi, tappa per tappa, a raccogliere indicazione o oggetti; se saltate qualche passaggio potreste sbagliare strada o peggio ancora arrivare a destinazione e vedervi annullare l'arrivo perchè non avete tutti gli oggetti che dovevate raccogliere.
A questo punto qualcuno potrebbe insorgere: "Ma ci sono già i computer che si occupano di tutte queste cose. Perchè devo perdere tempo a studiare il funzionamento dei protocolli?"
E' vero. Ci sono già i computer che si occupano di questo. Giusta considerazione. E allora perchè studiarsi i protocolli?
Alla fine dei conti i computer elaborano i dati utilizzando le regole dei protocolli. Ora, se fossi io a elaborare i dati, sempre rispettando le regole dei protocolli? Avrei la possibilità di forgiare pacchetti (così si chiamano i dati che circolano in rete) ad hoc. Crearli per soddisfare specifiche esigenze.
Qualche esempio:
Lo smurfing. Vi siete mai chiesti cosa succede quando utilizzate utility come Vortex, Smurf2k o Papasmurf? Viene CREATO un pacchetto ICMP (o UDP) ad hoc contenete l'indirizzo IP del tipo che dobbiamo bastonare. Non potendo mettere le mani(o non sapendo come farlo) nei pacchetti questo non sarebbe fattibile.
Alziamoci di qualche gradino e analizziamo qualcosa di più fine e sofisticato: Nmap.
Nmap, validissimo port scanner per casa *nix, tra le varie opzioni di cui si trova corredato permette di effettuare scansioni così dette STEALTH. Questo è possibile perchè vengono costruiti pacchetti casalinghi che permettono di cominciare una connessione ma non di terminarla, togliendo al target la possibilità di loggare la connessione (non può loggare un evento che non è avvenuto). Da qui il nome stealth.
O ancora Hping2, sempre per *nix, che permette di pingare un host inviando pacchetti costruiti seguendo le indicazioni dell'utente.
Le possibili applicazioni sono tante, basta usare la fantasia.
Il discorso è MOOOOOOOOOOOOOOOLTO più complesso delle poche righe scritte fino ad ora, ma spero di avervi fatto afferrare almeno il concetto.
Da qui in poi cercherò di spiegare in maniera (schifosamente) semplice il funzionamento di questa "via" obbligata e più avanti cercherò di parlare dei protocolli veri e propri, descrivendo dove possibile, almeno nei casi più importanti, anche come sono costituiti e come funzionano.

LA "VIA" OBBLIGATA

LO STACK OSI/ISO

Come detto prima, i dati, dal momento che gli inviamo alla circolazione vera e propria in rete, devono seguire una strada obbligata detta Stack.
Lo stack (letteralmente PILA) è una successione di strati, in cui i dati passano per essere codificati secondo il formato standard di comunicazione.
Il primo (e quello da cui poi nasce tutto il resto) stack è l' OSI (Open System Interconnect) di ISO (International Standard Organization). ISO, organizzazione mondiale che si occupa di definire gli standard nel campo dell'elaborazione, aveva strutturata un'architettura di rete che si appoggiava su sette strati:

Strato 1: FISICO

La strato 1 è il puro e semplice hardware usato per la circolazione delle informazioni, quindi, antenne satelliti e qualsiasi altro mezzo usato per tale scopo.

Strato 2: COLLEGAMENTO DEI DATI

Anche in questo caso abbiamo a che fare con l'hardware, e nello specifico, con le tecnologie Ethernet e Token Ring. Infatti è a questo livello che i dati vengono "spezzettati" per poi essere inviati al livello 1.

Strato 3: RETE

E qui entra in ballo il primo vero Protocollo: il protocollo IP.
Il protocollo IP si occupa di indirizzare i dati verso la giusta destinazione, scegliendo eventualmente la strada migliore da percorrere.

Strato 4: TRASPORTO

Su una lettera posso anche specificare 100 indirizzi, ma se nessuno si prende la briga di recapitarla questa non si muove. Stessa cosa vale per i dati in rete. Per ovviare a questo problema è stato creato lo strato di Trasporto, che implementando i protocolli di trasporto (TCP e UDP), si occupa della transizione dei dati cercando di farli arrivare tutti e senza errori.

Strato 5: SESSIONE

Lo strato sessione di occupa di coordinare una connessione tra due computer. Senza una sessione (e quindi una connessione) due computer non potranno MAI neanche tentare di spostare dati l'uno verso l'altro.

Strato 6: PRESENTAZIONE

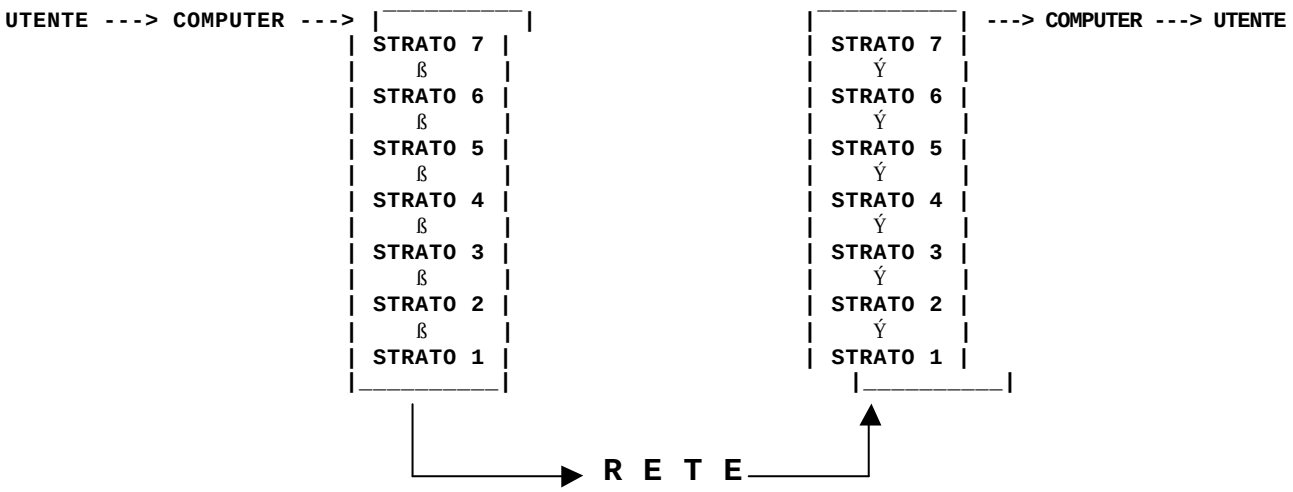
Questo strato interagisce direttamente col sistema operativo e col file system e si occupa di codificare i dati in movimento al fine di garantire la compatibilità anche fra macchine diverse. Senza di questo, le connessioni sarebbero limitate a macchine dello stesso tipo.

Strato 7: APPLICAZIONE

Detta brevemente, questo strato corrisponde all'interfaccia grafica (o a caratteri) del PC. Senza di questa non potreste inviare dati e il computer non saprebbe come visualizzare quelli che riceve.

Da una rapida analisi notiamo quindi che lo strato più basso corrisponde al puro hardware e quello più alto all'interfaccia utente.

Ricapitolando:



Ultima nota/considerazione:
l'architettura di rete OSI/ISO, nonostante la completezza e la sicura affidabilità è caduta in disuso ed è stata sostituita per due semplici motivi:
1) L'implementazione di questa architettura è estremamente complicata;
2) Quando OSI cominciò a imporsi come standard, il governo americano stava già usando lo stack TCP/IP per altro già testato su ARPANET. Mettendo a confronto le due architetture si resero conto che la seconda era molto più semplice da usare e di pari completezza. Quindi, soddisfatti, la adottarono e ne permisero la diffusione come standard.

LO STACK TCP/IP

Lo stack TCP/IP è l'architettura di rete maggiormente usata a livello mondiale e si evolve regolarmente con l'inserimento di nuovi protocolli, l'eliminazione di quelli obsoleti e le continue migliorie a quelli esistenti.
Concettualmente i le due architetture sono simili; anche TCP/IP adotta il concetto della stratificazione, ma in maniera alleggerita, in quanto utilizza 5 strati:

STRATO 1: FISICO

Come in OSI, ci si riferisce ai mezzi che i dati usano per "correre" (antenne, satelliti ecc.)

STRATO 2: COLLEGAMENTO DATI

Come sopra; si tratta del mezzo fisico con cui i dati vengono inviati.

STRATO 3: INTERNET

Analogo allo strato RETE di ISO; i dati vengono indirizzati.

STRATO 4: TRASPORTO

Anche qua vengono utilizzati i protocolli TCP e UDP per il trasporto dei dati.

STRATO 5: APPLICAZIONE
Questo è lo strato più complesso, in quanto svolge i compiti di SESSIONE, PRESENTAZIONE e APPLICAZIONE che in OSI venivano svolti rispettivamente dagli strati 5, 6 e 7.

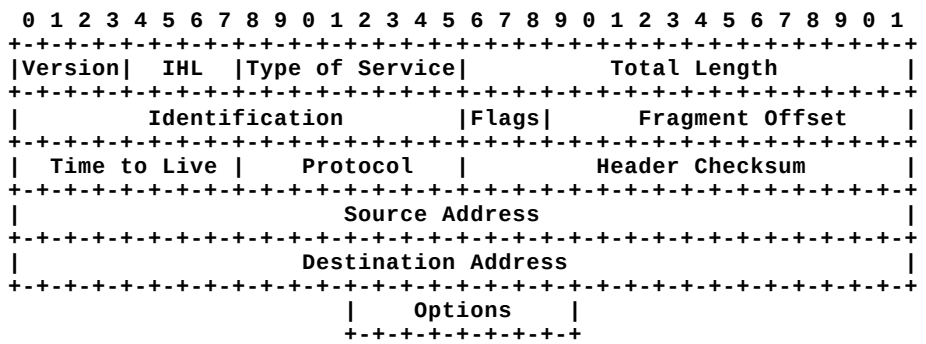
Bene, con questo concludo il discorso sul DOVE agiscono i protocolli. Spero almeno di aver dato una infarinatura minima sui concetti, perchè sicuramente qualcosa di quanto detto verrà ripreso più avanti.
Ed ora buttiamoci nella polpa della guida:

I PROTOCOLLI

Qua cercherò di elencare e descrivere per quanto mi è possibile i protocolli utilizzati dallo stack TCP/IP: I protocolli che tratterò sono:
IP, Internet Protocol;
TCP, Trasmission Control Protocol;
UDP, User Datagram Protocol;
ARP, Address Resolution Protocol;
RARP, Reverse Address Resolution Protocol;
ICMP, Internet Control Message Protocol;
FTP, File Transfer Protocol;
Telnet;
TFTP, Trivial File Transfer Protocol;
SNMP, Simple Network Management Protocol;
SMTP, Simple Mail Transfer Protocol;
POP3, Post Office Protocol v3;
IMAP4, Internet Message Access Protocol v4;
LDAP, Lightweight Directory Access Protocol;
NTP, Network Time Protocol;
HTTP, HyperText Transfer Protocol;
S-HTTP, Secure HyperText Transfer Protocol;
BOOTP, Boot Protocol;
RIP, OSPF, protocolli per i gateway e i router;
PPTP, Point to Point Tunnelling Protocol;
DHCP, Dynamic Host Configuration Protocol;
SSL, Secure Socket Layer;
SET, Secure Electronic Transaction;
IPSec, IP Security Protocol.

INTERNET PROTOCOL

L'Internet Protocol è il protocollo utilizzato per fornire ai dati un indirizzo di destinazione, ed eventualmente selezionare il miglior percorso di transito. Nonostante la sua utilità ha comunque delle limitazioni, perchè pur indirizzando i pacchetti non è in grado di garantirne l'arrivo a destinazione. Il paccheto IP viene chiamato datagramma, cioè una unità di dati autocontenente che utilizza un tipo di trasmissione differente dal byte - stream (flusso di dati).
Un header IP è così composto:



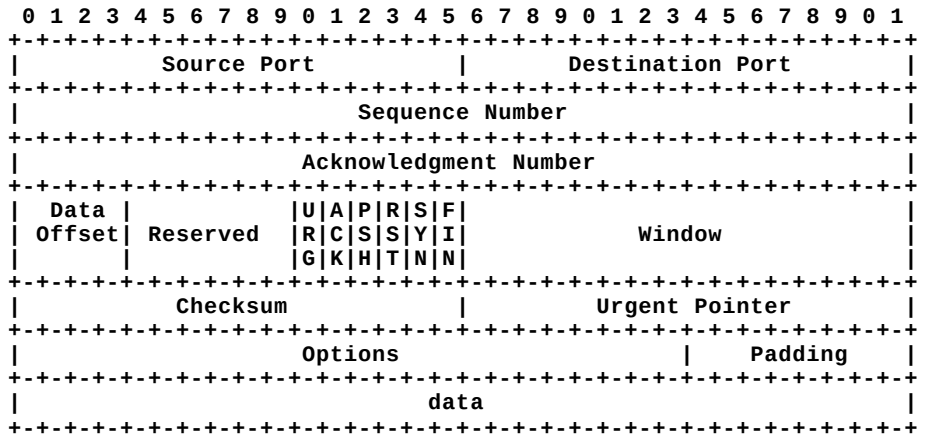
Version: (4 bit) Indica la versione dell'IP in uso;
IHL: (4 bit) Indica la lunghezza dell'header IP;
Type of Service: (8 bit) Usato per stabilire la priorità di consegna dei dati, è suddiviso a sua volta in
Precedence: Con un valore da 0 a 7 specifica l'importaza dei dati;

Delay: Se impostato indica di minimizzare i ritardi;
Throughput: Indica che si vuole massimizzare il throughput;
Reliability: Indica che si vuole massimizzare l'affidabilità;
Cost: Indica che si vuole massimizzare il costo.

Total Lenght: (16 bit) Specifica la lunghezza totale del pacchetto IP;
Identification: (16 bit) permette di identificare il pacchetto quando viene frammentato;
Flags: (3 bit)Utilizzato per il controllo della frammentazione;
Fragment Offset: (13 bit) Indica la distanza o il punto di rottura dall'inizio del datagramma nel caso questo sia frammentato. Per la ricostruzione del pacchetto, IP utilizza questo campo.
Time To Live: (8 bit) Indica il tempo di "vita" di ogni pacchetto. Questo valore viene decrementato a ogni passaggio in un router, e nel caso il router
riceva un pacchetto con TTL scaduto, lo distruggerà avvisando l'host mittente.
Protocol: (8 bit) Indica quale protocollo ha creato il pacchetto trasportato da IP. Ad esempio se il protocollo è ICMP avrà valore 1, se IGMP avrà valore 2, se TCP 6, se UDP 17 e così via;
Header Checksum: (16 bit) E' usato per la verifica dell'integrità dell'header IP;
Source Address: (32 bit) Specifica l'idirizzo della macchina che ha inviato il datagramma;
Destination address: (32 bit) Specifica l'indirizzo della macchina che deve ricevere il datagramma;
Options: (8 bit) Questo campo è usato per testare e correggere le applicazioni di rete.

TRASMISSION CONTROL PROTOCOL

Il TCP è il protocollo che viene usato per trasportare effettivamente i dati. Un pacchetto di questo tipo è diviso principalmente in due parti:
l'header del TCP che specifica tutti i parametri per il trasferimento;
i dati veri e propri da trasmettere.
Le principali "doti" che caratterizzano il TCP e che lo rendono estremamente affidabile sono:
1) Il TCP richiede che per il trasferimento sia stabilita una connessione tra le due macchine;
2) Ogni pacchetto inviato viene "etichettato" con una sequenza nuerica; questo per evitare che i dati, all'arrivo vengano riassemblati in maniera sbagliata;
3) Non è la macchina mittente che decide quali byte inviare, ma è la macchina destinataria che richiede in maniera esplicita la sequenza di byte giusta.
Questo è possibile in quanto TCP in ogni pacchetto specifica la lunghezza e il primo byte della sequenza inviata.
Un header TCP è così composto:



Source Port: (16 bit) Indica la porta usata dalla macchina che invia i dati;
Destination Port: (16 bit) Indica la porta sulla quale la macchina ricevente riceverà i dati;
Sequence Number: (32 bit) Indica l'ordine di riassemblaggio dei pacchetti;
Acknowledgment Number: (32 bit) Indica il numero della sequenze che il computer ricevente dovrebbe ricevere;
Data Offset: (4 bit) Indica la lunghezza dell'header TCP;
Reserved: (6 bit) Normalmente impostato a 0, è stato implementato per utilizzi futuri;
URG: (1 bit) Flag di controllo dell'urgenza dei dati;
ACK: (1 bit) Flag di controllo del Acknowledgment Number. Se è 1 il numero è valido;
PSH: (1 bit) Flag del controllo di "lancio". Se è 1, il pacchetto deve essere inviato immediatamente;
RST: (1 bit) Flag di reset. Se è 1 la connessione viene reinizializzata;
SYN: (1 bit) Flag di sincronizzazione che se impostato a 1 chiede all'host ricevente di sincronizzare i propri numeri di sequenza;
FIN: (1 bit) Flag di fine invio. Se impostato a 1 viene comunicato che non ci sono più dati da trasmettere;
Window: (16 bit) Indica il numero di byte che l'host accetterà;
Checksum: (16 bit) Campo che permette all'host ricevente di effettuare controlli sui dati;
Urgent Pointer: (16 bit) Puntatore all'ultimo byte di dati urgenti;
Options: Campo di lunghezza variabile che contiene alcune opzione del pacchetto;
Padding: Campo a lunghezza variabile usato per assicurarsi che l'header TCP termini entro i 32 bit;
Data: I dati reali da trasmettere.

Qualche esempio per chiarire le idee:

-Modalità di connessione Three-way handshake (connessione in tre passi)

1: SYN=1 ACK=0 SEQ-NUM=x CLIENT------------------>SERVER
La richiesta di connessione viene effettuata inviando un pacchetto con flag SYN impostato a 1, flag ACK a 0 e e Sequence number con valore x.

2: SYN=1 ACK=1 SEQ-NUM=y ACK-NUM=x+1 CLIENT<------------------SERVER
Il server chiamato risponde con flag SYN a 1, flag ACK a 1, Sequence Number con valore y e Acknowledge Number col precedente valore x incrementato di 1.

3: SYN=0 ACK=1 SEQ-NUM=x+1 ACK-NUM=y+1 CLIENT------------------->SERVER
Il chiamante a sua volta risponde con flag SYN a 0, flag ACK a 1, Sequence number col precedente valore x incrementato di 1 e Acknowledge Number corrispondente al precedente y ancora incrementato di 1.

CONNESSIONE STABILITA

-Modalità di trasferimento dei dati:
La modalità di trasferimento dei dati è analoga a quella per la richiesta di connessione.

1: SEQ-NUM=1500 ACK-NUM=5500 DATA=150
A--->B
A invia a B 150 byte di dati, in cui il primo ha come numero di sequenza 1500. Inoltra lo informa che ha ricevuto con successo i byte fino al 5499 e che attende un pacchetto di dati il cui numero di sequenza dovrà essere 5500.

2: SEQ-NUM=5500 ACK-NUM=1650 DATA=300
B----------->A
B risponde con la sequenza 5500 richiesta da A e conferma l'arrivo di byte fino a 1649(l'ACK-NUM conferma la ricezione dei byte indicati meno 1). A sua volta invia 300 byte.

3: SEQ-NUM=1650 ACK-NUM=5800 DATA=250
A----------->B
A invia la sequenza 1650; conferma la ricezione dei byte fino al 5799 aspettandosi la sequenza 5800. Invia 250 byte.

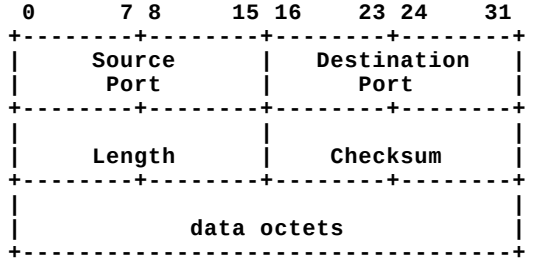
4: SEQ-NUM=5800 ACK-NUM=1900 DATA=0
B----------->A
B risponde con la sequenza 5800 e conferma la ricezione dei byte fino al 1899. Non contenendo dati, questo pacchetto non verrà confermato.

USER DATAGRAM PROTOCOL

Un' altro protocollo che lavora al livello di trasporto è UDP. Anche se fondamentalmente eseguono lo stesso lavoro, fra TCP e UDP esistono delle differenze fondamentali:

-UDP non richiede una connessione attiva;
-UDP è più veloce;
-UDP non lavora in byte-stream (flusso di dati);
-UDP non pone controlli sui dati inviati.

Uno dei pregi di UDP, è la possiblità di inviare messaggi in broadcasting e multicasting.
Detto in parole semplice UDP non è un protocollo di trasporto affidabile, in quanto i trasferimenti dei dati sono affidati alla "buona sorte", al contrario di TCP che numera e controlla tutti pacchetti trasmessi e ricevuti.
Pertanto, ogni applicazione che voglia utilizzare questo sistema di trasferimento, dovrà anche occuparsi di controllare i dati tramite i propri mezzi.



Source Port:(16 bit) La porta che verrà usata dall'host mittente per inviare i dati
Destination Port: (16 bit) La porta che verrà usata dall'host destinatario per ricevere i dati.
Lenght: (16 bit) La lunghezza dell'header UDP
Checksum: (16 bit) Usato per il controllo della validità dell'header.
Data Octets: (32 bit) I dati che verranno trasmessi

ADDRESS RESOLUTION PROTOCOL

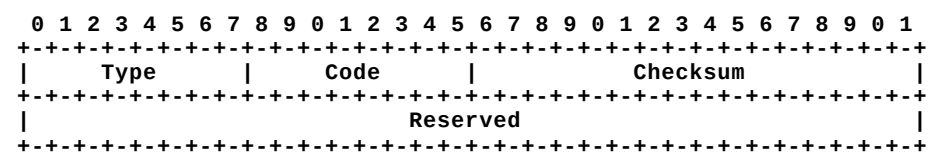
L' ARP è una richiesta di tipo broadcast che viene eseguita su una rete ethernet quando vogliamo comunicare con un computer di cui conosciamo solo l'indirizzo IP. La richiesta viene fatta contemporaneamente a ogni computer presente, fino a quando quello interessato non risponderà con il suo indirizzo fisico. Fatto questo, il suo indirizzo fisico verrà aggiunto nella ARP-Table del nostro PC, evitando così di dover ripetere la richiesta ogni volta che dobbiamo comunicare con quella macchina.

REVERSE ADDRESS RESOLUTION PROTOCOL

Antenato del DHCP, svolge la funzione esattamente contraria a quella del protocollo ARP, cioè scopre l'indirizzo IP partendo da un indirizzo MAC (fisico). Questo serviva quando bisognava utilizzare macchine prive di memorie di massa che a ogni riavvio non potendo disporre del proprio indirizzo IP, per il semplice fatto che non potevano imprimerlo da nessuna parte, al contrario del proprio indirizzo fisico che veniva registrato nella memoria ROM.

INTERNET CONTROL MESSAGE PROTOCOL

L'ICMP è un protocollo usato per il trasporto dei messaggi di errore e per il trasporto dei messaggi di servizio. Opera allo stesso livello di IP, anche lui si serve di IP per l'indirizzamento dei dati; proprio come IP non garantisce l'arrivo dei messaggi a destinazione.
ICMP è stato creato perchè IP non utilizzava sistemi di notifica dei messaggi di errore che potevano verificarsi durante il trasferimento dei dati. Ha comunque le sue limitazioni, in quanto non è in grado di "correggere" l'errore verificatosi, e comunque non può comunicarlo ai router intermedi, ma solo all'host che ha generato il pacchetto. Una particolarità: i messagi di errore creati sono progettati per non creare a loro volta altri messaggi di errore, onde da evitari i classici "broadcast storm".
L'header ICMP:



Type: (8 bit) Indica il tipo di messaggio da notificare (vedi più avanti);
Code: (8 bit) Indica il codice di errore per ogni tipo (vedi più avanti);
Checksum: (16 bit) Usato per controllare la lavidità dell'header;
Reserved: (32 bit) Usato se il tipo di messaggio lo richiede.

Tipi di messaggio:

Tipo 0 (echo reply)
Messaggio di richiesta
Descrizione: Usato in risposta all'ICMP tipo 8
Tipo 3 (Destination Unreachable)
Messaggio di errore
Descrizione: Viene generato da un router nel caso la macchina la macchina richiesta sia irraggiungibile; può essere creato anche dalla macchina destinataria nel caso venga richiesto un servizio non disponibile.
Tipo 4 (Source quench)
Messaggio di errore
Descrizione: Viene creato dai router che non riescono a gestire i messaggi in arrivo.

Tipo 5 (Redirect)
Messaggio di errore
Descrizione: Viene generato dai router e contiene l'indirizzo IP del router che dovrebbe essere contattato al posto di questo.
Tipo 8 (Echo)
Messaggio di richiesta
Descrizione: E' utilizzato per controllare la presenza di un computer in rete; è generato dal comando Ping.
Tipo 9 (Router Advertisement)
Messaggio di richiesta
Descrizione: E' utilizzato dai router per segnalare la propria presenza in rete.
Tipo 10 (Router solicitation)
Messaggio di richiesta
Descrizione: Forza i router in ascolto a identificarsi.
Tipo 11 (Time exceeded)
Messaggio di errore
Descrizione: Viene generato quando il campo Time To Live (TTL) è 0.
Tipo 12 (Parameter problem)
Messaggio di errore
Descrizione: Indica un problema nei parametri dell'header.
Tipo 13 (Time Stamp request)
Messaggio di richiesta
Descrizione: Usato per misurare le prestazioni e il debugging.
Tipo 14 (Time Stamp Reply)
Messaggio di richiesta
Descrizione: Risposta all'ICMP tipo 13
Tipo 15
----------NON PIU' USATO----------
Tipo 16
----------NON PIU' USATO----------
Tipo 17 (Address mask request)
Messaggio di richiesta
Descrizione: Viene usate per le richieste RARP per scoprire la maschera subnet usata dalle macchine in rete.
Tipo 18 (Address mask reply)
Messaggio di richiesta
Descrizione: Risposta all'ICMP tipo 17

Codici per i tipi di messaggio:

-Tipo 3

Codice 0: Rete irrangiungibile
Codice 1: Host irraggiungibile
Codice 2: Protocollo Irraggiugibile
Codice 3: Porta irragiugibile
Codice 4: Necessaria frammentazione
Codice 5: Tragitto di origine fallito
Codice 6: Rete di destinazione sconosciuta
Codice 7: Host di destinazione sconosciuto
Codice 8: Host di origine isolato
Codice 9: Rete di destinazione vietata dallamministrazione
Codice 10: Host di destinazione vietato dallamministrazione
Codice 11: Rete irraggiungibile per il tipo di servizio
Codice 12: Host irraggiungibile per il tipo di servizio
Codice 13: Comunicazione proibita dall'amministrazione
Codice 14: Violazione della precedenza da parte dell'host
Codice 15: Scorciatoia in vigore

-Tipo 5

Codice 0: Reindirizzamento per rete
Codice 1: Reindirizzamento per host
Codice 2: Reindirizzamento per tipo di servizio e rete
Codice 3: Reindirizzamento per tipo di servizio e host

File Transfer Protocol

FTP è un protocollo di "alto" livello, infatti opera nello strato Application dello stack. Come la maggior parte dei protocolli che operano in quello strato non ha una struttura propria nel vero senso della parola, è più che altro un insieme di comandi e funzioni creati e implementati per lo svolgimento di particolari servizi. In questo caso, FTP, si occupa del trasferimento dei file.
Per poter utilizzare il servizio FTP (che deriva dal nome del protocollo che usa), bisogna connettersi alla porta 21 dell'host che lo offre, attendere il banner di benvenuto e quindi loggarsi all'interno del sistema tramite una procedura simile a questa:

[Server] 220 FTPDaemon-Server version x.xx.xx
[Client] USER tuousername
[Server] 331 Password required for tuousername.
[Server] PASS tuapassword
[server] 230 User username logged in.
[Client] ....

I numeri accanto ai messaggi del server sono riportati sotto.
FTP utilizza due porte per la trasmissione: la porta 20 per il passaggio dei parametri inerenti il trasferimento (indirizzo IP, porta, modalità) e la 21 per il trasferimento vero e proprio dei dati.
La trasmissione può avvenire in due modi:
ATTIVA: Trasmessi i parametri, il server rimane in ascolto e aspetta che il client faccia le richieste;
PASSIVA: E' il client che si mette in ascolto, lasciando al server la facoltà di "gestire" la sessione; ad esempio è utile nel caso il server sia dietro a un firewall che può rendere difficoltosa la connessione da parte del client.

Comandi e sintassi:

USER
sintassi: **USER <SP> <username> <CRLF>**
Specifica il nome dell'account con il quale ci si loggerà nel sistema

PASSWORD
sintassi:**PASS <SP> <password> <CRLF**>
Specifica la password relativa al nome acoount utilizzato

ACCOUNT
sintassi:**ACCT <SP> <account-information> <CRLF>**
Usato per identificare l'account utente utilizzato.

CHANGE WORKING DIRECTORY
sintassi:**CWD <SP> <pathname> <CRLF>**
E' l'equivalente del comando CD di DOS e Linux.

CHANGE TO PARENT DIRECTORY
sintassi:**CDUP <CRLF>**
Torna immediatamente alla directory principale.

STRUCTURE MOUNT
sintassi:**SMNT <SP> <pathname> <CRLF>**
E' l'equivalente del comando MOUNT di Linux.

REINITIALIZE
sintassi:**REIN <CRLF>**
Reinizializza una sessione utente.

LOGOUT
sintassi:**QUIT <CRLF>**
Termina una sessione utente

DATA PORT
sintassi:**PORT <SP> <host-port> <CRLF>**
Permette di specificare un indirizzo e una porta diverse da quelle utilizzate di defaul durante la sessione

PASSIVE
sintassi:**PASV <CRLF>**
Attiva la trasmissione in modalità passiva.

REPRESENTATION TYPE
sintassi: **TYPE <SP> <type-code> <CRLF>**
Comando per il cambio di formato.

FILE STRUCTURE
sintassi: **STRU <SP> <structure-code> <CRLF>**
Specifica la modalità di invio dei dati.

TRANSFER MODE
sintassi: **MODE <SP> <mode-code> <CRLF>**
Questo comando specifica il modo per spedire i dati.

RETRIEVE
sintassi: **RETR <SP> <pathname> <CRLF>**
Questo comando permette di trasferire una copia del file specificato.

STORE
sintassi: **STOR <SP> <pathname> <CRLF>**
Questo comando permette di creare un file sul server.

STORE UNIQUE
Sintassi: **STOU <CRLF>**
E’ simile a STORE, ma il file viene creato nella directory corrente

APPEND
sintassi:**APPE <SP> <pathname> <CRLF>**
Crea un file, e se il file già esiste si limita ad aggiungere i dati.

ALLOCATE
sintassi: **ALLO <SP> <decimal-integer> [<SP> R <SP> <decimal-integer>] <CRLF>**
Serve per garantire spazio sufficiente ad un file che deve essere creato.

RESTART
Sintassi: **REST <SP> <marker> <CRLF>**
Reinizialliza il file trasferito.

RENAME FROM
sintassi: **RNFR <SP> <pathname> <CRLF>**
Questo comando specifica il vecchio percorso di un file che deve essere rinominato.

RENAME TO
sintassi: **RNTO <SP> <pathname> <CRLF>**
Rinomina il file.

ABORT
sintassi: **ABOR <CRLF>**
Pemette l'aborto delle azioni in corso.

DELETE
sintassi: **DELE <SP> <pathname> <CRLF>**
Cancella il file specificato

REMOVE DIRECTORY
sintassi: **RMD <SP> <pathname> <CRLF>**
Cancella la directory specificata

MAKE DIRECTORY
sintassi: **MKD <SP> <pathname> <CRLF>**
Crea la directory specificata

PRINT WORKING DIRECTORY
sintassi: **PWD <CRLF>**
Visualizza il percorso della directory

LIST
sintassi: **LIST [<SP> <pathname>] <CRLF>**
Visualizza i file e le directory presenti

NAME LIST
sintassi: **NLST [<SP> <pathname>] <CRLF>**
Specifica i file in una directory presente nell'argomento.

SITE PARAMETERS
sintassi: **SITE <SP> <string> <CRLF>**
Specifica particolari attività del server.

SYSTEM
sintassi: **SYST <CRLF>**
E’ usato per scoprire che tipo di sistema gira sul server.

STATUS
sintassi: **STAT [<SP> <pathname>] <CRLF>**
Indica lo stato del trasferimento del file in corso.

HELP
sintassi: **HELP [<SP> <string>] <CRLF>**
Visualizza l’help. Può essere usato singolarmente o con un comando come parametro.

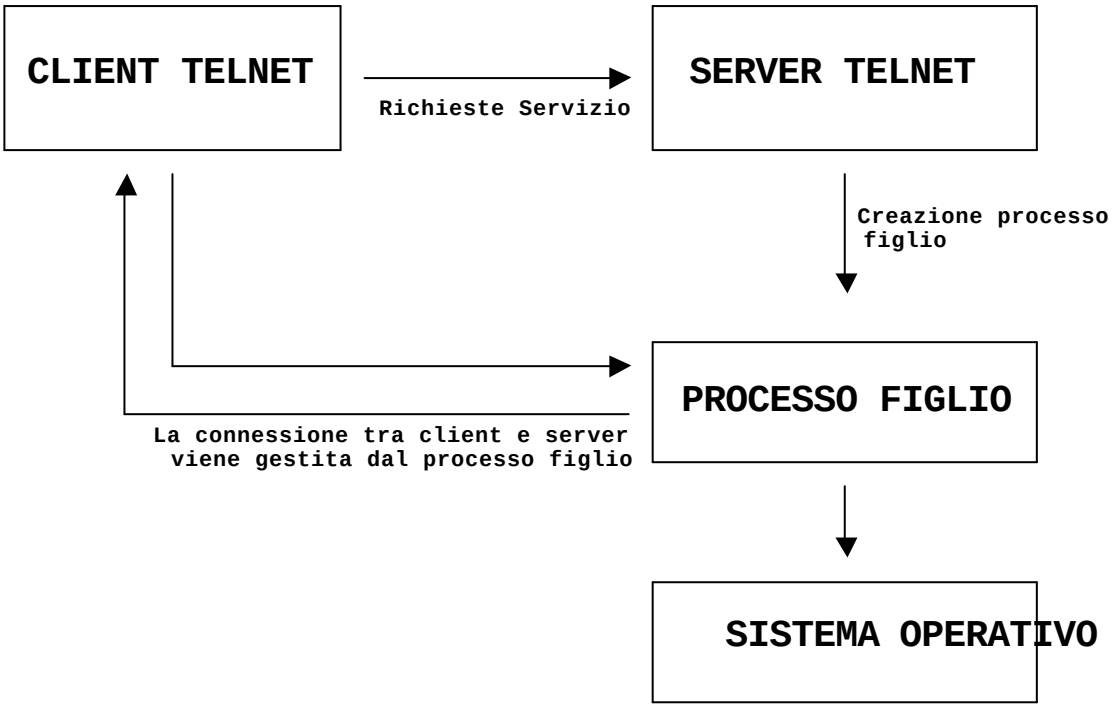
NOOP
sintassi: **NOOP <CRLF>**
Eseguito questo comando non potranno più essere eseguite operazioni.

Codici di errore del server:

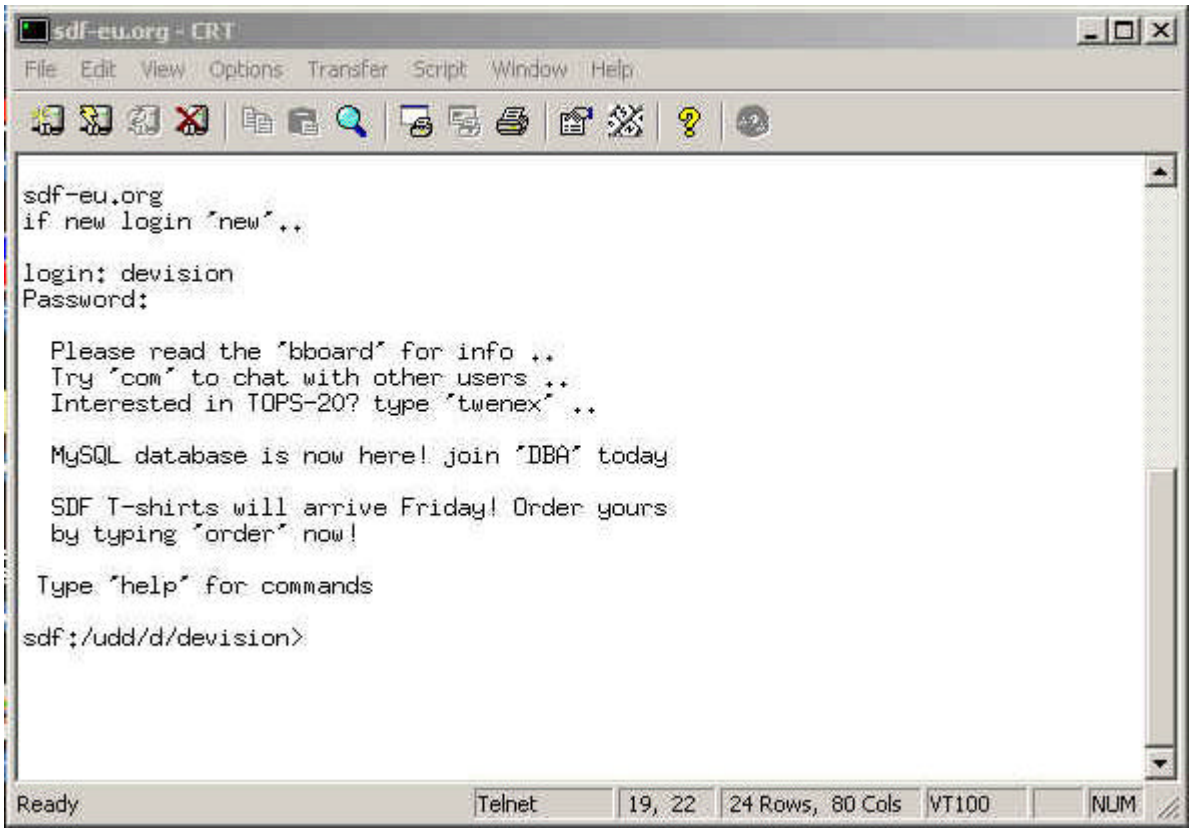
200 Commando di okay.
202 Commando non implementato, superfluo in questa zona.
211 Stato di sistema, o sistema in replica di un help.
212 Stato della directory.
213 Stato del file.
214 Messaggio di Help.
215 NAME system type.
220 Servizio pronto per un nuovo utente.
221 Servizio di controllo di chiusura della connessione.
225 Connessioni Dati aperta; nessuna trasmissione in progressione.
226 Chiusa connessione dati.
227 Entra nella modalità passiva (h1,h2,h3,h4,p1,p2).
230 Utente loggato, procedere.
250 Richiesta azione file okay, completa.
257 "PATHNAME" creata.
331 Nome utente okay, necessaria password.
332 Necessaria azione per il login.
350 Richiesta azione su file, attendere altre informazioni.
421 Service non avviabile, chiusura della connessione.
425 Non poter aprire una connessione dati.
426 Connessione chiusa; trasferimento abortito.
450 Richiesta azione su file non presa.
451 Richiesta azione abbortita: errore locale in processione.
452 Richiesta di azione non presa / Spazio disponibile insufficiente.
500 Errore di sintassi, comando non individuato.
501 Errore di sintassi nei parametri o nell'argomento.
502 Comando non implementato.
503 Cattiva sequenza dei comandi.
504 Comando non implementato per questi parametri.
530 Non loggato.
532 Bisogno di un account per copiare i files.
550 Richiesta di azione non presa. (file non trovato)
551 Richiesta di azione abbortita: tipo pagina sconosciuto.
552 Richiesta azione su file abbortita.
553 Richiesta di azione non presa.

TELNET

Telnet è un protocollo (e un programma) utilizzato per effettuare login da remoto. Vuol dire che, dal proprio computer è possibile utilizzare delle macchine non fisicamente presenti come se fossero queste le macchine che stiamo utilizzando. Telnet è presente di default nella maggior parte dei sistemi che utilizzano TCP/IP come windows o unix e per utilizzarlo generalmente basta lanciare il comando Telnet dal prompt dei comandi, seguito dal nome host o dall’indirizzo IP della macchina a cui vogliamo connetterci. Come impostazione predefinita, il server telnet accetta connessioni sulla porta 23. La gestione della connessione da parte del server è particolare, in quanto non è il programma server propriamente detto a eseguire questa funzione: nel momento in cui la connessione viene accettata, viene creato una sorta di processo “figlio” a cui verrà passata la trasmissione dei dati; sarà questo processo a far interagire il client telnet con il sistema operativo contattato. Un esempio di sessione telnet:



Ecco come appare ai nostri occhi una sessione Telnet:



La trasmissione dei caratteri si basa sul concetto standard di NVT (Network Virtual Terminal); questo per evitare che sorgano delle incompatibilità durante l'invio di sequenze di escare, la cui interpretazione dipende normalmente dal sistema operativo (ad esempio il ritorno a capo in alcuni è CR-LF, in altri LF e in altri ancora CR). Per la comunicazione possono essere usati tutti i 128 caratteri stampabili e una serie di caratteri non stampabili, riconosciute dal demone Telnet come vere sequenze di comandi:

Codice: 240 Nome comando: SE
Fine del negoziato per l’attivazione dei parametri opzionali.

Codice: 241 Nome comando: NOP
Nessuna operazione.

Codice: 242 Nome comando: DATA MARK
Viene inviato in combinazione con il flag Urgent Data di TCP e avvisa il server Telnet di analizzare il flusso di dati alla ricerca di comandi che richiedono inoltro immediato.

Codice: 243 Nome comando: BREAK
Standard NVT del comando BREAK.

Codice: 244 Nome comando: INTERRUPT PROCESS
Interruzione del processo.

Codice: 245 Nome comando: ABORT OUTPUT
E’ l’interruzione dell'output di un programma.

Codice: 246 Nome comando: ARE YOU THERE
Dice al client di dare un segno di “vita” in modo da confermare la propria presenza.

Codice: 247 Nome comando: ERASE CHARACTERS
Cancella l’ultimo carattere inviato.

Codice: 248 Nome comando: ERASE LINE
Cancella l’ultima linea inviata.

Codice: 249 Nome comando: GO AHEAD
Viene inviato quando un processo non può proseguire in mancanza di input.

Codice: 250 Nome comando: SB
Inizio del negoziato per l’attivazione dei parametri opzionali.

Codice: 251 Nome comando: WILL
-Se inviato come richiesta:
E’ stata attivata localmente un’opzione e chiede che venga attivata anche dall’altra parte.
-Se inviato come risposta:
E’ stato chiesto se è possibile attivare un’opzione. WILL è la risposta affermativa.

Codice: 252 Nome comando: WONT
E’ stato chiesto se è possibile attivare un’opzione. WONT è la risposta negativa.

Codice: 253 Nome comando: DO
-Se inviato come richiesta:
Chiede se è possibile attivare un’opzione.
-Se inviato come risposta:
L’opzione attivata in remoto è confermata anche su questo sistema.

Codice: 254 Nome comando: DONT
L’opzione attivata in remoto è stata rifiutata.

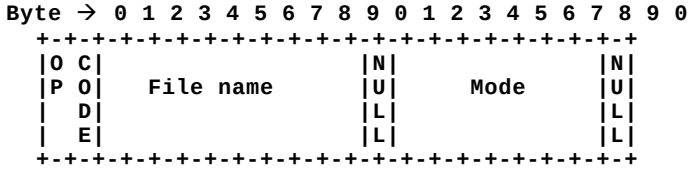
TRIVIAL FILE TRANSFER PROTOCOL

Il Tftp è un protocollo simile al protocollo FTP, ma molto limitato in quanto permette solo la lettura/srcittura di file e mail da un server ad un altro. Non è possibile listare il contenuto di una directory e non è prevista nessuna procedura di autentificazione dell'utente.
Il protocollo Tftp utilizza il protocollo UDP per il trasferimento dei dati.
Le modalità di trasferimento utilizzabili sono:
netascii, che definisce una codifica dei byte in caratteri ASCII a 8 bit, invece che 7 bit secondo lo USASCII;
octet, che corrisponde al modo binario delle versioni precedenti, caratterizzato da byte di 8 bit generici;
mail, che corrispondono a caratteri netascii, spediti da un utente.

I pacchetti manipolati da Tftp sono:
RRQ (Richiesta di Lettura)
WRQ (Richiesta di scrittura)
DATA (invio di un blocco)
ACK (notifica)
ERROR (Errore)

Non esiste una procedura vera e propria per stabilire una connessione. Il server che fa richiesta di limiterà a inviare un pacchetto RRQ nel caso sia intenzionato a leggere e un WRQ nel caso sia intenzionato a scrivere.

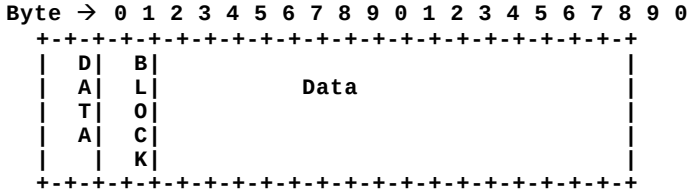
Pacchetti RRQ/WRQ:



Op Code: (2 byte) Contiene il codice operativo del pacchetto (RRQ = 1, WRQ = 2);
File name: (lunghezza variabile) contiene il nome del file da trasferire in formato netascii;
Null: (1 byte) Serve per terminare il campo File Name;
Mode: (lunghezza variabile) Indica una delle tre modalità di trasferimento;
Null: (1 byte) Serve per terminare il campo Mode;

I pacchetti RRQ e WRQ vengono usati rispettivamente per le richieste di lettura e scrittura su un server.

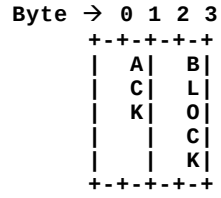
Pacchetti DATA:



Data: (2 byte) Contiene il codice operativo del pacchetto (3);
Block: (2 byte) E' il numero del pacchetto inviato;
Data: (lunghezza variabile fino a 512 byte) Sono i dati veri e propri trasmessi;

Il pacchetto DATA è il pacchetto che contiene i veri dati da trasmettere.

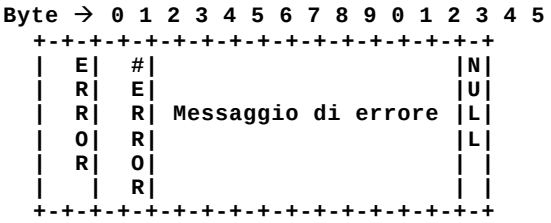
Pacchetto Ack:



Ack: (2 byte) Codice 4
Block: (lunghezza variabile) Contiene lo stesso numero del pacchetto Data di cui questo è la notifica.

Viene utilizzato per notificare la ricezione di un pacchetto.

Pacchetto Error:

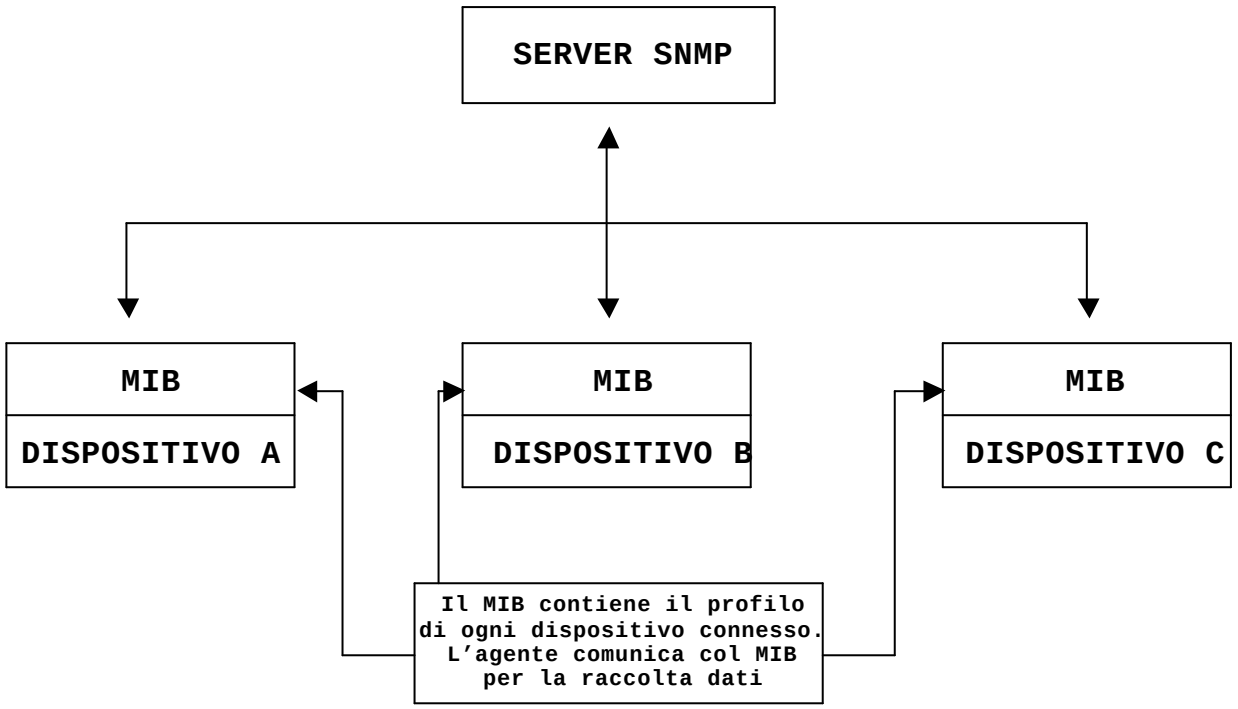


Error: (2 byte) Codice operativo 5
#Error: (2 byte) Codice dell' errore
Messaggio di errore: (lunghezza variabile): Contiene una breve descrizione dell'errore
Null: (1 byte) Carattere di terminazione.

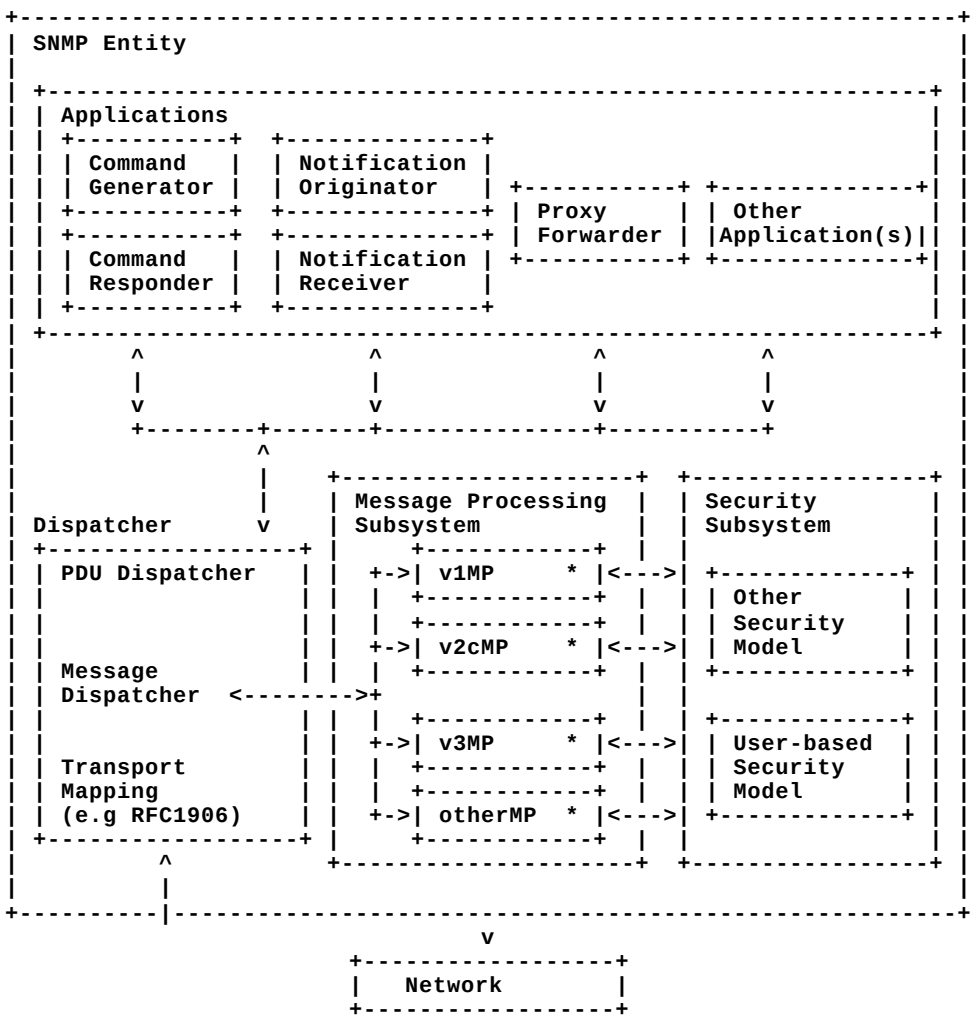
Tftp viene generalmente usato su stazioni senza disco fisso o da Terminali X per il trasferimento di immagini di disco, files e font.
Altre applicazioni non sono possibili in quanto ogni aspetto della sicurezza riguardante Tftp deve essere curato dall'amministratore di sistema.

Simple Network Management Protocol

SNMP è un protocollo che lavora al livello di applicazione. E' utilizzato per la gestione della rete, in quanto è in grado di raccogliere tutti i dati riguardanti le periferiche o qualsiasi cosa connessa alla rete, valutarne le prestazioni, notificare problemi e errori ed eventualmente aiutare nello sviluppo della rete stessa l'amministratore di sistema.
SNMP utilizza tre componenti per il proprio funzionamento:
i nodi gestiti;
l'agente:
l'NMS, ovvero il sistema di gestione.
I nodi gestiti sono router, hub, bridges, stampanti e qualsiasi altro mezzo che possa comunicare all'esterno della rete; l'agente è un software che si occupa di raccogliere le informazioni e di renderle disponibili a SNMP; l'NMS è il sistema di gestione della rete, il sistema che si occupa di monitorare le prestazioni e gestire le risorse richieste.
Le comunicazioni tra NMS e dispositivi avviene tramite i MIB, una sorta di dichiarazione di variabile a disposizione di ogni componente, che tengono traccia di tutte le sue caratteristiche.



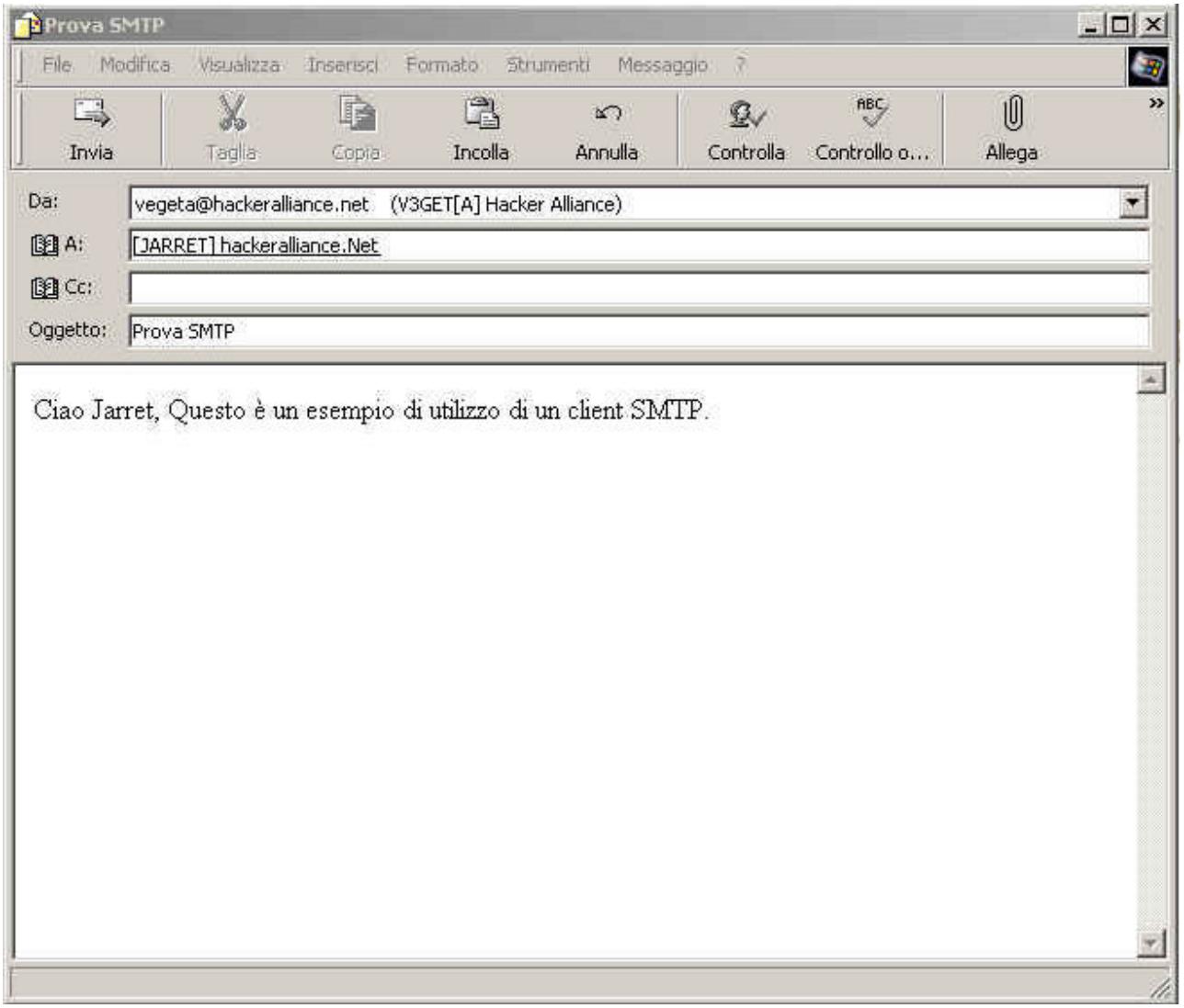
Esempio schematico dell'entità SNMP, ovvero dell'insieme dei processi che simultaneamente lavorano su una stazione SNMP:



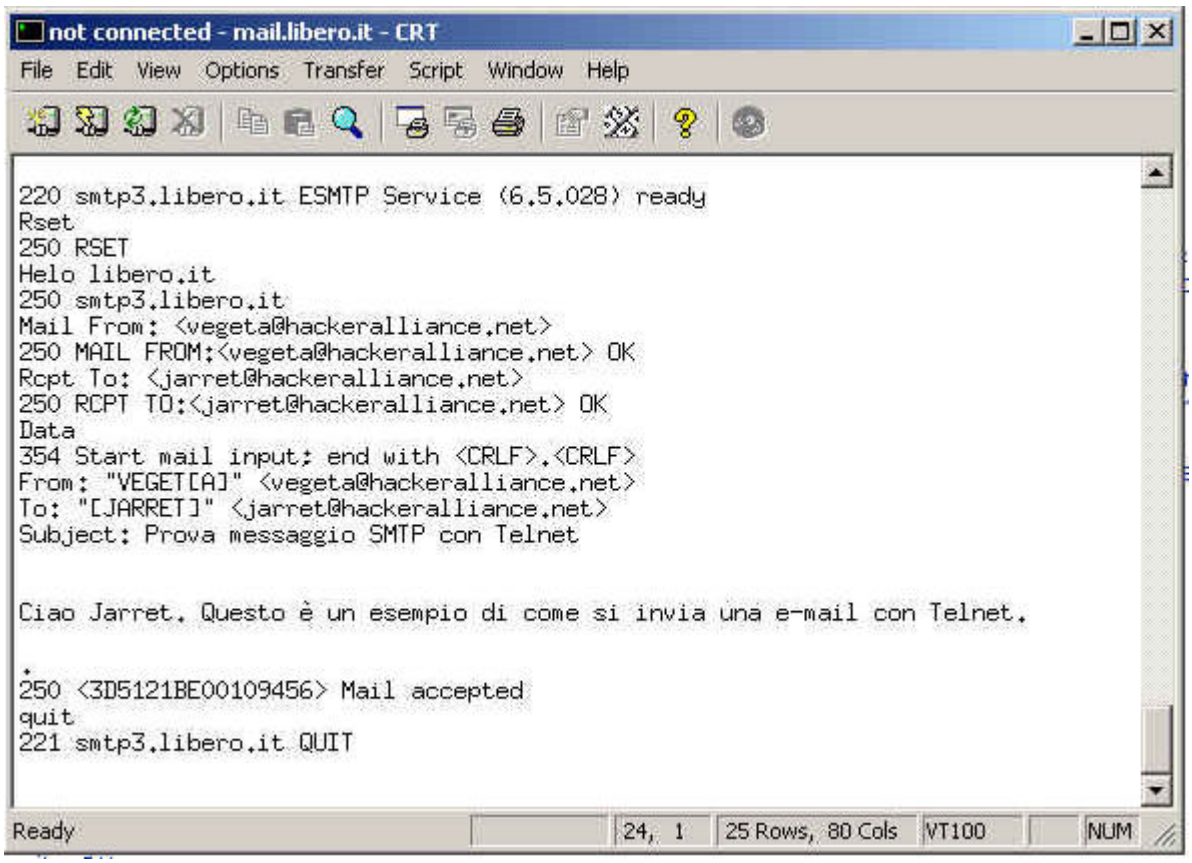
L’accesso agli oggetti MIB può essere di lettura/scrittura o sola lettura, e viene gestito da uno schema di autentificazione, che garantisce l’ autenticità dei messaggi SNMP.

Simple Mail Transfer Protocol

Il Simple Mail Transfer Protocol è un protocollo che lavora al livello applicazione ed è quello che si occupa di gestire la posta che inviamo.
Di default la porta del servizio SMTP è la 25.
Nonostante esistano molti client (Outlook, Eudora, Incredimail ecc.) che si occupano della gestione del servizio, questo può essere utilizzato anche “manualmente”, magari utilizzando un semplice client Telnet per connettersi e seguire le procedure stabilite per l’invio di una mail; questo è possibile perché i comandi utilizzati sono in chiaro e di facile comprensione/utilizzo.
Nei due casi apparirebbe così:



Invio di e-mail con Outlook;



invio di e-mail tramite client Telnet.
Codici di errore e comandi sono descritti di seguito.

Comandi e sintassi:

HELLO

sintassi: **HELO <SP> nome_provider.it /.net/.org/.com ecc. <CRLF>**
Appena connessi, il server si identificherà con banner.
Il client risponde col comando HELO seguito dal nome provider con cui siamo connessi.

MAIL

sintassi: **MAIL FROM: <SP> <indirizzo_mail_mittente@provider> <CRLF>**
All' inizio della transizione, il comando MAIL indica l'indirizzo del mittente.

RECIPIENT

sintassi: **RCPT TO: <SP> <indirizzo_mail_destinatario@provider> <CRLF>**
Indica l'indirizzo del destinatario (recipient = ricevente).

DATA

sintassi: **DATA <CRLF>**
Dopo questo comando, il server riconoscerà tutto quello che viene scritto come testo della mail. Per indicare la fine della mail, al termine del corpo, si dovrà inviare la sequenza <CRLF>.<CRLF>:

```
corpo della mail            ←--- corpo della mail + invio
                            ←--- <CRLF> (invio)
.                           ←--- punto (.)
                            ←--- <CRLF> (invio)

Mail accepted               ←--- risposta del server
```

SEND
sintassi: **SEND <SP> FROM:<reverse-path> <CRLF>**
Invia una mail a uno o più terminali.

SEND OR MAIL
sintassi: **SOML <SP> FROM:<reverse-path> <CRLF>**
Serve per spedire un messaggio ad un terminale, o, se questo non è attivo alla sua mail-box.

SEND AND MAIL
sintassi: **SAML <SP> FROM:<reverse-path> <CRLF>**
Invia la mail sia al terminale che alla mail-box.

RESET
sintassi: **RSET <CRLF>**
Resetta la transizione, svuota i buffer e permette di ricominciare da capo.

VERIFY
sintassi: **VRFY <CRLF>**
Verifica la presenza di una mail.

EXPAND
sintassi: **EXPN <SP> <string> <CRLF>**

HELP
sintassi: **HELP [<SP> <string>] <CRLF>**
Restituisce un aiuto sul comando passato come parametro.

NOOP
sintassi: **NOOP <CRLF>**
Indica di non poter effettuare nessuna operazione.

QUIT
sintassi: **QUIT <CRLF>**
Indica la fine della transizione.

TURN
sintassi: **TURN <CRLF>**
Indica al server di invertire i ruoli. Se il server accetta, lui stesso diventerà un client SMTP.

Codici di errore del server:

211 System status, or system help reply
214 messaggio di Help
220 <domain> Servizio pronto
221 <domain> Servizio ha chiuso il canale
250 Richieste di azione completata, OK.
251 Utente non locale; si spedirà a <forward-path>
354 Inizia l'input dei dati; finisce con <CRLF>.<CRLF>
421 Servizio non avviabile
450 Richiesta di azione non avviabile [E.g., mail-box piena]
451 Richiesta di azione abortita: errore locale durante il processo
452 Richiesta di azione abortita: spazio di sistema insufficiente
500 Errore di sintassi, comando non riconosciuto
501 Errore di sintassi nei parametri o negli argomenti
502 Comando non implementato
503 Cattiva sequenza del comando
504 Parametri del comando non implementati
550 Richiesta di azione non presa: mail-box inavviabile
551 Utente non locale; per favore prova <forward-path>

552 Richiesta di azione abortita: si eccede l'allocazione di spazio
553 Richiesta di azione non presa: nome della mail-box non permesso
554 Transazione fallita

Post Office Protocol v3

Come SMTP, POP3 è un protocollo che si occupa di gestire la posta; in questo caso viene gestita quella che preleviamo da un server. Anche qua è possibile utilizzare un client di posta o utilizzare il servizio manualmente tramite un semplice client Telnet.
Di default la porta utilizzata da questo servizio è la 110.
Una cosa che lo distingue dal protocollo SMTP è il fatto di dover utilizzare una username e una password per poter accedere al servizio.

I comandi del servizio sono:

USER
sintassi: **USER <SP> username <CRLF>**
All' inizio della transizione, il comando USER indica il nome utente con il quale si vuole accedere.

PASS
sintassi: **PASS <SP> password >CRLF>**
Segue sempre il comando USER e indica la password relativa al nome utente utilizzato.

QUIT
sintassi: **QUIT <CRLF>**
Termina la sessione di lavoro.

STAT
sintassi: **STAT <CRLF>**
Chiede informazioni sullo stato della posta. Il server risponderà indicando il numero dei messagi presenti e la loro dimensione totale.

LIST
sintassi: **LIST <SP> numero messaggio <CRLF>**
Se inviato singolarmente restituisce l'output del comando STAT per ogni messaggio presente. Se inviato con un parametro, restituisce l'identificativo del messaggio seguito dalla sua dimensione in ottetti.

RETR
sintassi: **RETR <SP> numero messaggio <CRLF>**
Chiede che vanga visualizzato il messaggio richiesto.

DELE
sintassi: **DELE <SP> numero messaggio <CRLF>**
Marca il messaggio definito per la cancellazione. La cancellazione effettiva verrà eseguita al successivo aggiornamento.

Gli messaggi restituiti dal server possono **+OK** in caso di esecuzione riuscita e **–ERR** in caso di esecuzione fallita.

Esempio di sessione POP3:
[Server]+OK Server POP3 version x.xx
[Client] USER yyy
[Server] +OK (La casella yyy esiste)
[Client] PASS
[Server] +OK 3 (messaggi in casella)
[Client] STAT
[Server] +OK 3 15467 (numeri messagi presenti e peso totale)
[Client] RETR 3
[Server] +OK 670 (spedisce il primo messaggio)

```
[Server] .
[Client] DELE 2
[Server] +OK (il secondo messaggio è stato cancellato)
[Client] QUIT
[Server] +OK
```

Il resto dei protocolli che ho annunciato li tratterò in una seconda parte del documento. Per il momento sorbitevi questo.
La storia è la stessa; per commenti, suggerimenti, chiarimenti, dubbi, insulti e madonne, qua:
vegeta@hackeralliance.net