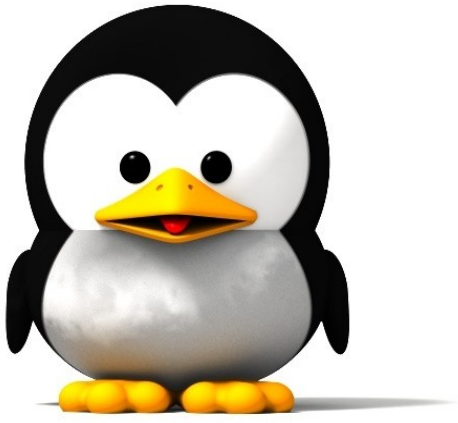
ESAME DI STATO 2008
CECCARELLI FRANCESCO – V A INFORMATICA

GNU/LINUX E LE RETI TCP/IP

ISTITUTO TECNICO INDUSTRIALE STATALE
ITIS
E. MATTEI URBINO

via L. Pacioli, 22 - 61029, Urbino
Tel. 0722-328021 Fax 0722-320179
E-mail: itis@emattei-urbino.it
Sito internet: www.emattei-urbino.it



INDICE

PARTE I
Il software libero

Al giorno d'oggi abbiamo quasi tutti a che fare, in molteplici situazioni, con il software libero. Talvolta anche inconsapevolmente. Penso però che pochi sappiano cos'è e cosa caratterizza il software libero. E' perciò necessaria una premessa generale, che ci faccia capire bene di cosa stiamo parlando, prima di addentrarci nei meandri del maggior esponente di questa vera e propria corrente filosofica dell'informatica odierna: il sistema operativo GNU/Linux.

FILOSOFIA

Quando una persona compra o scarica un programma proprietario essa non lo possiede veramente: non è infatti di sua proprietà ma gli viene concesso in licenza d'utilizzo, facendo in modo che prima di installare il programma egli accetti un'EULA (End User License Agreement). Tutto ciò indipendentemente dal costo del programma: dalle piccole tool freeware all'ultima versione di Adobe Photoshop.
Quando invece una persona ottiene una copia di un software libero, ottiene non solo il programma compilato ma anche i suoi codici sorgenti. In questo modo diventa lui stesso il vero proprietario del programma, con la possibilità di modificare, redistribuire e utilizzare il programma stesso per qualsiasi scopo. Ma attenzione, questo, come molti potrebbero pensare, non costringe schiere e schiere di programmatori a lavorare gratis. Il lavoro va pagato. Libero infatti non vuol dire gratis, come molti pensano. Molti programmi liberi sono gratuiti in versione ridotta, ma devono essere regolarmente acquistati se si vuole accedere alle versioni cosiddette Enterprise. Per altri invece è gratuito il download del software da Internet, mentre se si vogliono ottenere i CD originali, completi di assistenza a lungo termine, si deve regolarmente pagare. L'unico vincolo che viene imposto sul software libero è che se un ipotetico acquirente dovesse ottenere un programma di questo genere, dovesse modificarlo e poi redistribuirlo, egli deve per forza continuare a rispettare il canone della licenza libera, segnalando questo all'autore.
La licenza per eccellenza dei programmi liberi è la GNU General Public License (GPL), redatta proprio dalla famosa associazione di Richard Stallman, pioniere del concetto di copyleft, ovvero la facoltà di una persona di copiare e redistribuire liberamente il programma, in opposizione al concetto di copyright. La GNU GPL garantisce infatti le cinque libertà fondamentali che il software libero deve rispettare per fregiarsi di questo nome:

- *libertà 0*: libertà di eseguire il programma, per qualsiasi scopo;
- *libertà 1*: libertà di studiare come funziona il programma e adattarlo alle proprie necessità;
- *libertà 2*: libertà di ridistribuire copie;
- *libertà 3*: libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio.

Queste libertà sono alla base della GPL e hanno sempre e comunque come prerequisito fondamentale quello della distribuzione del programma assieme ai sorgenti; solo questo sistema inoltre permette di essere i veri proprietari del programma acquistato, lasciando comunque all'autore la legittima facoltà di trarre guadagno dal proprio lavoro. Ed oggi i programmi che adottano questo tipo di licenza sono veramente tanti, e la loro presenza sta crescendo esponenzialmente, anche a causa dei molteplici errori commessi dalle grandi case di software proprietario. Mi riferisco per esempio a programmi come Mozilla Firefox: infatti quest'ultimo in pochi anni ha conquistato più di un quarto

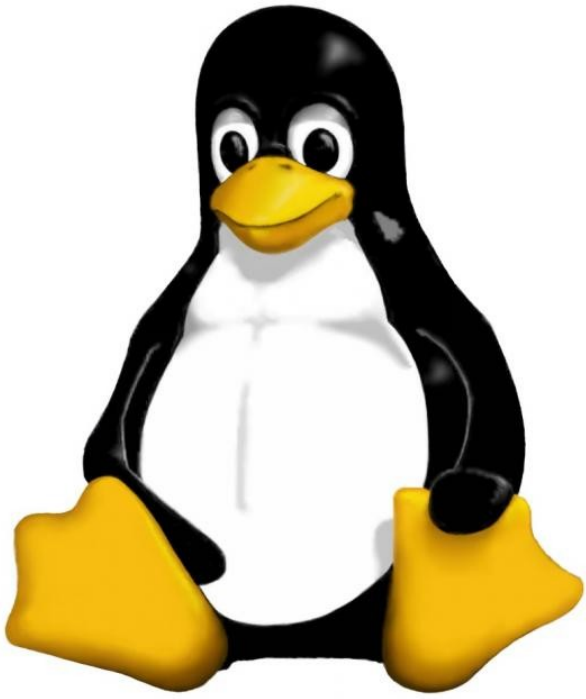
del mercato europeo, forte della sua licenza libera, che gli permette una maggiore invulnerabilità a bug critici e di sicurezza (ricordo che infatti i sorgenti essendo liberi possono essere visionati da tutti e gli errori possono essere più facilmente individuati e corretti), una compatibilità e leggerezza senza confronti (la Mozilla Foundation non ha infatti stipulato nessun accordo con nessuna casa produttrice di hard disk) e il vantaggio di essere completamente gratuito, a meno di volontarie donazioni. Inoltre questo modello libero continua a stupire ovunque: ricordo la diffusione capillare del CMS Wordpress e l'oramai indispensabile fonte di sapere che rappresenta Wikipedia, in cui tutti posso accedere e scrivere articoli per il bene di tutti, e dove è praticamente impossibile assistere ad errori, involontari e non: infatti questi possono durare solo pochi minuti, prima che qualcuno si accorga e sostituisca tutto con le giuste informazioni. Infine, per ultimo, e' importante citare anche la piattaforma Apache-PHP, che al giorno d'oggi è la più utilizzata al mondo, a discapito del rivale Microsoft IIS e della nuovissima piattaforma .NET.
Per concludere il modello del software libero è ormai un fenomeno che gli informatici moderni non possono ignorare: ma il suo successo non è dovuto solo all'apparente costo zero che offre, ma anche alla maggiore sicurezza e stabilità, diretta conseguenza dei principi GPL, che le case proprietarie si rifiutano di offrire. Chiaro, ancora i vecchi “standard de facto” come il sistema operativo Microsoft Windows e la suite Microsoft Office risultano gli indiscriminati dominatori del mercato. Ma non è possibile sottovalutare un modo di concepire l'informatica come quello dei programmi liberi, che stanno piano piano crescendo e conquistando sempre più PC.

STORIA

Fino agli anni '70 i personal computer non esistevano e l'informatica era cosa solo per i ricercatori universitari. Non esisteva ancora la differenza tra software proprietari e non, poiché tutti i programmatori collaboravano scambiandosi i propri sorgenti.
Dagli anni '80 invece, con l'avvento dei primi PC, lo sviluppo software passò nelle mani delle aziende, che cominciarono a produrre software per PC: nacquero così i primi software proprietari, che impedivano agli utenti di utilizzare come meglio credevano i loro programmi. Nel frattempo però, Richard Stallman, che lavorava al Massachusetts Institute of Technology (MIT), si accorse di questo fenomeno, e decise, nel 1985, di istituire la Free Software Foundation (FSF), al fine di eliminare le restrizioni sulla copia, sulla redistribuzione, sulla comprensione, sulla modifica dei programmi e, soprattutto, di portare avanti il progetto GNU (GNu's Not Unix). Quest'ultimo prevedeva lo sviluppo di un sistema operativo completamente libero e capace di sostituire in tutto e per tutto qualsiasi altro sistema e programma proprietario: già nel 1990 il progetto poteva vantare infatti un editor di testo (Emacs), un compilatore (GCC), un debugger (GDB) e varie librerie per la programmazione in C. Ma mancava ancora la parte fondamentale del sistema operativo: il kernel. I lavori per il kernel in realtà erano già cominciati nel 1986, ma a dire il vero ancora oggi il nucleo di GNU (chiamato HURD) non è ancora pronto ad essere utilizzato.
Parallelamente a tutto ciò uno studente dell'università di Helsinki, Linus Torvalds, si era ormai stancato dell'incompatibilità del sistema operativo Minix con il suo computer, che montava il nuovo processore Intel 80386 a 32 bit.

Decise così, un po' per studio, un po' per gioco, di sviluppare un nuovo kernel, che lui prima chiamò Freax, ma che poi si tramutò in Linux. Così il giovane programmatore decise di informarsi sugli standard POSIX, pensando che il suo nuovo kernel doveva divenire Unix-like, ma che doveva avere una licenza che permettesse ai suoi utenti di modificarne e adattarne il codice come meglio volevano. Torvalds così, quando decise di caricare su un server FTP pubblico il codice dei piccoli ed instabili moduli da lui sviluppati, distribuì i propri sorgenti sotto la licenza GNU GPL, riuscendo ad ottenere l'aiuto di programmatori da tutto il mondo. Essi ottimizzarono il kernel per applicativi GNU e vi associarono la shell BASH: nacque così il primo sistema operativo GNU completo, che prese il nome di GNU/Linux.
Le release cominciarono poi ad avvicendarsi in fretta e nel 1994 ne uscì finalmente la prima versione stabile. Inoltre, il progetto fu proprio in questo periodo inglobato da diverse aziende di software, che, grazie alla licenza GPL, decisero di portare avanti progetti propri, senza però modificare i termini di licenza: nacquero così le prime distribuzioni.

PARTE II
Il sistema operativo GNU/Linux



Abbiamo, nel capitolo precedente, introdotto il concetto di distribuzione, ma ancora non l'abbiamo ufficialmente definito. Una distribuzione è un sistema operativo completo di kernel Linux, software libero GNU, documentazioni, manuali, un sistema di installazione guidato per utenti meno esperti e talvolta anche software commerciale. Esse vengono per lo più distribuite gratis via Internet e hanno permesso l'allargamento del bacino di utenza di GNU/Linux; prima infatti il sistema operativo era usato solo dai guru dell'informatica, mentre con l'avvento delle prime “distro” (come si chiamano in gergo) anche utenti meno competenti poterono avvicinarsi a GNU/Linux.
Al giorno d'oggi esistono quasi 300 distribuzioni differenti, per lo più nate da fork di altri progetti, che differiscono tra loro solo per il parco software che mettono a disposizione e per come lo gestiscono; ecco le principali:

- *Debian*: la distribuzione più vicina al progetto GNU, poiché composta soltanto da software libero. Essa è da sempre emblema di sicurezza e affidabilità, rappresentando insieme a Red Hat una delle due distribuzioni più usate ed antiche, da cui poi sono partiti la maggior parte degli altri progetti. Essa offre un ottimo sistema di gestione dei pacchetti software, detto APT (Advanced Packaging Tool), che attraverso un sistema di repository permette all'utente il download, l'installazione e la configurazione dei programmi di cui ha bisogno in pochi secondi;
- *Red Hat*: una delle poche distribuzioni che, anche se non da molto tempo, può essere ottenuta esclusivamente a pagamento. Solitamente non viene aggiornata alle ultime novità, preferendo versioni di kernel e componenti datate, ma stabili. Infine proprio Red Hat ha avuto il merito di realizzare il diffuso sistema di pacchetti RPM (Red Hat Package Manager), adottato poi da tutte le distribuzioni di sua derivazione;
- *Fedora*: la distribuzione gratuita sponsorizzata da Red Hat. Piuttosto curata nell'aspetto, viene aggiornata frequentemente con le ultime release dei software che offre. Il sistema di pacchettizzazione è basato naturalmente su RPM e l'installazione disponibile è una delle più user-friendly del settore;
- *Gentoo Linux*: distribuzione che permette di ottimizzare e rendere estremamente flessibile il sistema, attraverso un'installazione completamente personalizzabile. Non è adatta ad utenti alle prime armi, ma un punto a suo favore rimane comunque nella grande disponibilità della comunità che l'utilizza;
- *Mandriva*: conosciuta fino a poco tempo fa come Mandrake, è una tra le distribuzioni più diffuse e maggiormente orientate all'uso desktop. E' distribuita sia in forma gratuita che come prodotto commerciale, e deriva anch'essa dal progetto Red Hat;
- *Slackware*: la distribuzione più longeva e più utilizzata dai puristi del software libero, ma sconsigliata a coloro che non hanno tempo e voglia di imparare da capo ad usare un nuovo sistema operativo. Essa è caratterizzata da un sistema di gestione dei pacchetti molto rigido e la relativa scarsità di software precompilato può mettere in difficoltà l'utente alle prime armi. Risulta però adatta ad utenti esperti che conosco già le potenzialità di GNU/Linux e che hanno bisogno di un sistema privo di componenti inutili;
- *SuSE*: celebre distribuzione europea, molto usata a livello aziendale, ma rivolta anche all'utente desktop. Anch'essa basata su RPM, è un prodotto commercializzato da Novell, ditta molto discussa per il recente accordo con Microsoft sullo sviluppo di un progetto che mira all'interoperabilità tra Windows e la sua distro;

- *Ubuntu*: distribuzione derivata da Debian salita alla ribalta per la facilità d'installazione e d'utilizzo. Ne esistono numerose varianti, che si differenziano tra loro per i programmi che propongono: tra queste possiamo elencare Kubuntu, Xubuntu, Edubuntu e Medibuntu.

SHELL

Come abbiamo più volte ribadito Linux è un kernel. E niente più. Quindi da bravo kernel esso implementa tutte le funzioni di accesso all'hardware sottostante, in modo da rendere trasparente alle applicazioni ciò che succede durante gli accessi alle risorse. Tutto in un unico file oltretutto, poiché Linux è un kernel monolitico, a cui possono essere aggiunti dinamicamente moduli a caldo.
In particolare il kernel Linux si interfaccia con i programmi tramite system call, primitive che possono essere richiamate dagli altri software per accedere alle risorse. L'utente però interagisce con il kernel tramite un programma specifico, messo a disposizione dal sistema operativo stesso, che viene detto shell. La shell è in grado di eseguire i programmi applicativi e di permettere loro di accedere alle system call del kernel; essa si presenta con un'interfaccia non grafica, ma a caratteri, simile in apparenza a quella che presentava il buon vecchio DOS:

Questa è l'interfaccia che ci accoglie una volta che la fase di accensione e quella di login sono state completate. La notazione è la solita dei sistemi Unix:

<nome utente>@<nome host>: <cartella in cui ci si trova> { $ | # }

Per quanto riguarda invece l'ultimo carattere, siamo in presenza di un dollaro se le credenziali fornite nella fase di login sono quelle di un normale utente, mentre troviamo un cancelletto se l'utente con cui ci siamo loggati è l'amministratore. Vedremo in seguito le particolarità sugli utenti e sui diritti che questi hanno in GNU/Linux.
Esistono diverse implementazioni di shell per GNU/Linux, ma quella più usata è sicuramente la shell BASH. Essa può essere programmata mediante un linguaggio di scripting, che mette a disposizione del programmatore tutte le strutture di un qualsiasi altro linguaggio ad alto livello; non tratteremo però la sua programmazione.
Nella shell possiamo digitare i comandi nella seguente sintassi, che si presenta più o meno uguale per tutti:

<nome comando> <opzioni> <parametri>

seguito dal tasto Invio.

Ogni comando così dato in Linux non fa altro che richiamare un programma eseguibile presente, come vedremo, in cartelle apposite. Se vogliamo eseguire però un programma che non è presente in queste cartelle non dobbiamo specificarne solo il nome, ma anche il percorso completo in cui esso si trova all'interno del file system. Ad esempio se abbiamo compilato un programma *prova* e questo si trova nella cartella */home* allora dobbiamo digitare nella shell:

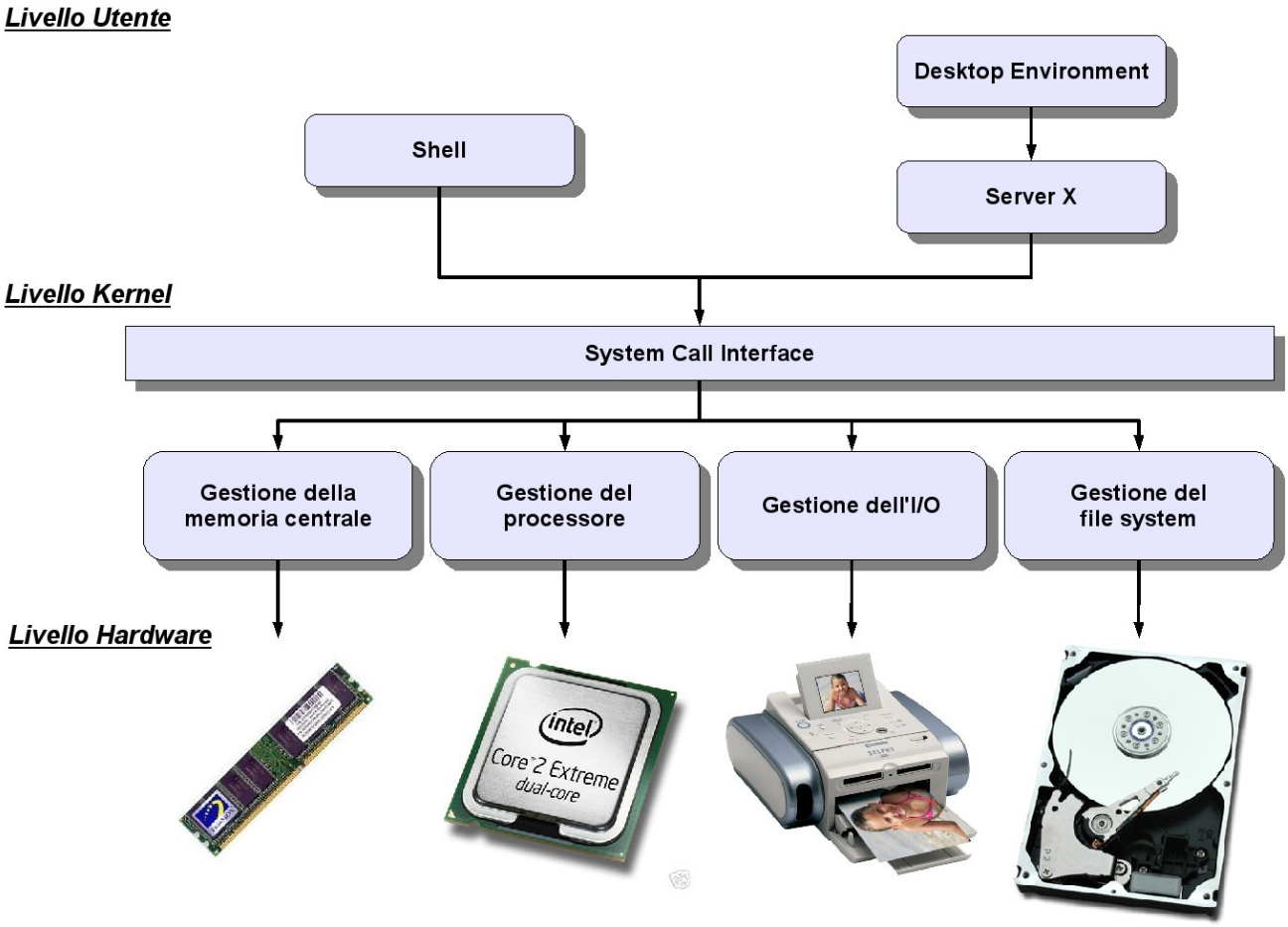
/home/prova

e quando spingeremo il tasto Invio questo verrà eseguito.
Possiamo inoltre specificare delle opzioni, ovvero quell'insieme di modi in cui può funzionare un programma. Esse quando sono specificate sono quasi sempre precedute da un – e spesso vogliono essere seguite da uno o più parametri.
Un parametro è un informazione che passiamo al programma per eseguire il suo lavoro. Può essere ad esempio il nome di un file su cui salvare l'output oppure, in un applicativo di rete, l'indirizzo di un computer.
I principali comandi della shell sono:

- *logout*, che esce dalla shell, tornando alla fase di login;
- *man <parametro>*, che visualizza la pagina di manuale del comando o del file di sistema specificato come parametro; contiene informazioni utili su come utilizzare programmi e loro file di configurazione;
- *pwd*, che restituisce il punto del file system in cui ci troviamo;
- *cd <cartella>*, che ci permette di muoverci all'interno del file system, specificando la cartella in cui vogliamo spostarci;
- *ls <cartella>*, che stampa a video il contenuto della cartella specificata. Se eseguito senza parametri stampa il contenuto della cartella corrente;
- *cp <file> <nuova posizione>*, che consente di copiare un file dalla posizione in cui ci troviamo ad un'altra; se vogliamo copiare un file che non si trova nella nostra posizione allora dobbiamo specificarne tutto il percorso;
- *mv <file> <nuova posizione>*, che funziona allo stesso modo del comando precedente, ma permette di spostare il file; se nel secondo parametro al posto di una cartella è specificato un nuovo nome il file viene rinominato;
- *rm <file>*, che permette di eliminare un file del sistema. Attenzione, i file rimossi non sono più recuperabili!
- *mkdir <nome>*, che consente all'utente di creare, dove ne ha i permessi, una nuova cartella col nome specificato;
- *rmdir <cartella>*, che elimina la cartella specificata nel parametro;
- *more <file di testo>*, che stampa a video il contenuto di un file ASCII;
- *vim <file di testo>*, che ci permette di editare o creare un file di testo, nel caso il nome specificato non esista.

Nelle distribuzioni orientate all'uso desktop però un'interfaccia a caratteri non può bastare. E' così stata sviluppata all'interno di ogni sistema un'architettura client-server che prevede un processo server, detto X, che viene avviato all'accensione del sistema e che fornisce tutti gli strumenti per disegnare finestre sullo schermo e accedere alle system call. Il server X però da solo non serve a molto. Ecco dunque che gli vengono affiancati dei client, ovvero un insieme di programmi che sfruttano i servizi messi a disposizione dal server X per creare un'interfaccia grafica (GUI). Questo insieme di programmi è detto Desktop Manager e trova le sue più famose implementazioni in KDE e GNOME.

Riassumiamo dunque con uno schema:



Nel proseguire dell'esposizione faremo però uso quasi esclusivamente dell'interfaccia a caratteri, poiché meglio si presta alla configurazione del sistema.

FILE SYSTEM

Il file system è quel particolare meccanismo con cui il sistema operativo organizza i dati sulle memorie di massa. In GNU/Linux il file system è gestito secondo lo standard Unix per cui *everything is a file*; ciò vuol dire che ogni periferica, ogni processo, e ogni qualsiasi altra risorsa è vista dal sistema operativo come un banale file. Per far ciò, a livello di kernel, ogni file system comunica con un'interfaccia software detta Virtual File System (VFS), che ha come unica cartella radice la / e ha il compito di rendere Linux indipendente dal reale tipo di file system a cui accediamo. Da un punto di vista logico, nella root del VFS verranno poi montati ricorsivamente, uno dentro l'altro, tutti i reali file system (ext2, nfs, fat32...) di ogni partizione, che saranno in seguito visibili, all'interno della radice, come sue sottocartelle.
Tipicamente GNU/Linux necessita di due partizioni per lavorare correttamente: una, non formattata e che quindi non viene montata, detta swap, al fine di allocare le pagine di memoria virtuale. L'altra, formattata per lo più con file system Ext3, che contiene invece tutte le subdirectory della radice. All'interno di essa verranno poi montate ricorsivamente anche tutte le partizioni contenute in chiavi usb, hard disk esterni, floppy, CD o altre partizioni dello stesso hard

disk interno.
Ecco qua le piu' importanti directory di un tipico file system di GNU/Linux; la partizione principale è montata nella cartella radice, che a sua volta contiene:

- *bin/*, per gli eseguibili di sistema;
- *boot/*, per i file che interessano il boot loader, immagine del kernel compresa;
- *dev/*, per i file che rappresentano le periferiche, le porte e gli altri dispositivi del PC;
- *etc/*, che contiene due parti fondamentali del sistema:
  - *gli script di avvio*, che contengono tutti i programmi in esecuzione automatica (demoni), necessari all'avvio dei vari servizi;
  - *i file di configurazione del sistema e dei programmi applicativi*: essi non sono scritti in un linguaggio universale, ma è importante sapere che in tutti possiamo decidere commenti anteponendo ad essi il carattere #, proprio come nei linguaggi di programmazione ad alto livello;
- *home/*, che contiene le subdirectory in cui sono montate le partizioni con i dati degli utenti;
- *lib/*, con le principali librerie e i moduli del kernel che si possono caricare in run-time;
- *mnt/*, che contiene le directory dove solitamente sono montati i supporti rimovibili;
- *proc/*, che contiene le immagini dei processi
- *root/*, per i file dell'utente amministratore;
- *sbin/*, per gli eseguibili riservati all'amministratore;
- *sys/*, per file necessari al corretto funzionamento del sistema operativo;
- *tmp/*, per i file temporanei, creati dai vari applicativi;
- *usr/*, per le librerie e gli eseguibili installati nel sistema;
- *var/*, per alcuni dati variabili, come cache e spool directory.

Inoltre Linux, contrariamente ad altri sistemi operativi concorrenti, è in grado di leggere sulla maggior parte dei file system in circolazione, garantendo la massima compatibilità fra file system tipici di sistemi operativi diversi.
Le partizioni degli hard disk interni sono solitamente montate in automatico all'avvio del sistema, poiché sono contenute nel file */etc/fstab*. Per tutte le altre però è necessario il montaggio e lo smontaggio logico, da effettuare manualmente. Il comando per montare partizioni è *mount*:

mount <device> <mount point>

dove dobbiamo specificare:

- *il dispositivo da montare*, che troviamo nella cartella */dev*;
- *il mount point*, che rappresenta la sottocartella in cui andrà montato logicamente il dispositivo;

Per smontare un dispositivo si utilizza invece il comando *umount*, nella sintassi:

umount <device>

Entrambi i comandi necessitano dei privilegi di root.

UTENTI E PERMESSI

Quando fu sviluppato Linux il mondo era molto diverso da oggi. Il PC non era lo standard assoluto che oggi conosciamo, ma tutta l'informatica del tempo ruotava attorno ai grandi microcomputer, la cui potenza era divisa fra più

terminali stupidi, che permettevano a più utenti di lavorare simultaneamente. GNU/Linux quindi, derivando da Unix, non poteva che essere un sistema operativo multiutente. Infatti in GNU/Linux esistono due tipi di utenti:

- *l'utente root*, che rappresenta l'amministratore e che possiede qualsiasi tipo di permesso su tutto il sistema; non è dunque consigliato utilizzare questo utente per la normale attività, ma solo per le operazioni di manutenzione, poiché un programma applicativo eseguito come root avrà la possibilità di accedere a tutti i file dell'hard disk, anche quelli di sistema;
- *tutti gli altri utenti*, che avranno privilegi inferiori a quello di root, ma che proprio per questo assicureranno una maggior protezione contro programmi con scopi malevoli.

Il primo problema che però si presenta allo sviluppatore di un sistema multiutente è certamente quello della gestione dei permessi di accesso alle risorse. I permessi possono essere di tre tipi:

- *in lettura* (r);
- *in scrittura* (w);
- *in esecuzione* (x).

Ogni file inoltre ha un proprietario e solo esso, assieme all'onnipotente utente root, può deciderne i relativi permessi da shell.
Oltretutto tutti gli utenti del sistema operativo sono organizzati in gruppi. Questo comporta l'aumento del numero di permessi che il sistema operativo deve gestire, che per ogni risorsa diventano:

- tre bit (rwx) per decidere i permessi che il proprietario ha su di essa. Infatti l'utente root se volesse potrebbe benissimo decidere di prendere controllo della risorsa e negargli l'accesso;
- tre bit (rwx) per i permessi che hanno gli altri componenti del gruppo a cui fa parte il proprietario;
- tre bit (rwx) per i permessi che hanno tutti gli altri.

Il comando usato per modificare i permessi di un file è *chmod*, che interpreta i permessi come una combinazione tra numeri ottali; in particolare valgono le seguenti corrispondenze:

NUMERO OTTALE	PERMESSI	DESCRIZIONE
0	`- - -`	Nessuna autorizzazione
1	`- - x`	Esecuzione
2	`- w -`	Scrittura
3	`- w x`	Scrittura ed esecuzione
4	`r - -`	Lettura
5	`r - x`	Lettura ed esecuzione
6	`r w -`	Lettura e scrittura
7	`r w x`	Tutti i permessi: lettura, scrittura ed esecuzione

La sintassi del comando è la seguente:

chmod <permessi> <nome file>

dove per permessi si intende una combinazione di tre numeri ottali, che esprimono in ordine i permessi del proprietario, del gruppo e di tutti gli altri.

Per assegnare ad un file un nuovo proprietario o un nuovo gruppo dobbiamo invece utilizzare il comando *chown*, nella sintassi:

chown <utente>:<gruppo> <nome file>

mentre per visualizzare permessi e proprietari dei file di una cartella utilizziamo il comando *ls*, affiancato dall'opzione *-l*.
Per quanto riguarda invece la gestione degli utenti e dei gruppi, l'utente root può:

- aggiungere un nuovo utente, con il comando *adduser <nome>*;
- rimuovere un utente, con *deluser <nome>*;
- aggiungere un nuovo gruppo, con il comando *addgroup <nome>*;
- rimuovere un gruppo, con *delgroup <nome>*.

Possiamo aggiungere invece utenti ai gruppi editando il file */etc/group* e accodando alla riga del gruppo gli utenti che vi vogliamo aggiungere, separati da una virgola. Se la modifica coinvolge un utente che ha già effettuato l'accesso al sistema, egli deve pero' necessariamente ripetere la procedura di login, se vuole che le modifiche siano applicate.
Inoltre ogni utente ha una password associata. Questa può essere modificata tramite il comando:

passwd <username>

Da ricordare che mentre l'utente root può cambiare qualsiasi password senza sapere la precedente, gli altri utenti possono cambiare solo la propria password e solo inserendo prima quella vecchia.
Abbiamo dunque descritto la gestione degli utenti in GNU/Linux: una gestione abbastanza semplice, ma che permette all'amministratore di sistema di rendere invulnerabile l'elaboratore da malintenzionati e da software indesiderati. Infatti senza disporre dei permessi di root un ipotetico virus potrà solo accedere alle risorse dell'utente che ha eseguito (anche inavvertitamente) il programma maligno, con il solo inconveniente della possibile perdita di dati; inconveniente irrisorio ai fini della stabilità del sistema.

GESTIONE DEL SOFTWARE

Una delle differenze chiave tra distribuzioni sta nella gestione dei pacchetti software installati. La distribuzione che utilizziamo per la dimostrazione pratica è Kubuntu, di derivazione Debian, perciò sarà qui descritto il sistema APT. APT è il gestore standard dei pacchetti in Kubuntu, scelto per merito della sua estrema semplicità di utilizzo: infatti esso sfrutta una serie di sorgenti HTTP e FTP (dette repository) che permettono all'utente di scaricare, installare e configurare software in modo automatico; gli indirizzi web dei repository sono specificati nel file */etc/apt/sources.list*, ma possono essere aggiornati e modificati a mano dall'utente; il comando principale di APT è *apt-get*:

apt-get <azione> <nome pacchetto>

dove azione corrisponde la maggior parte delle volte a una delle seguenti parole chiave:

- *install*: che serve a scaricare e ad installare il pacchetto specificato. Se necessario scarica e installa anche le dipendenze necessarie, ovvero quei pacchetti che e' obbligatorio installare con un programma;
- *remove*: usato per rimuovere il pacchetto specificato dal parametro;

- *update*: aggiorna gli elenchi dei pacchetti contenuti dai repository; è fondamentale usare questo comando subito dopo aver aggiunto nuovi repository al file *sources.list*, oppure per controllare la presenza di pacchetti aggiornati. Viene eseguito senza parametro aggiuntivo;
- *upgrade*: installa gli aggiornamenti che sono stati trovati con il comando *apt-get update*. Anch'esso viene eseguito senza specificare nessun altro parametro.

Importante è ricordare che per utilizzare il comando *apt-get* sono necessari i privilegi di amministratore.

PARTE III
Lo stack TCP/IP

Ora che abbiamo espresso le nozioni fondamentali riguardo l'uso del sistema operativo possiamo passare a descrivere come questo implementa e gestisce i protocolli di rete.
Una rete è un insieme di computer totalmente autonomi, ma che grazie ad un opportuno sistema di comunicazione sono in grado di scambiarsi messaggi e condividere risorse. Esistono diversi modelli per l'implementazione di una rete, ma il modello che descriveremo sarà quello più usato al giorno d'oggi: lo stack TCP/IP. Infatti anche se forse il modello più famoso è quello ISO/OSI sappiamo che quest'ultimo non è mai stato realmente implementato e per questo motivo la scelta migliore ricade sullo “standard de facto” di Internet e della maggior parte delle reti locali.
Questi due modelli hanno comunque la stessa struttura: entrambi considerano i computer della rete come un'architettura gerarchica a livelli (layer), dove ogni livello raggruppa servizi simili o strettamente correlati. I livelli sono disposti come in una pila (stack), in cui il livello sottostante fornisce servizi a quello superiore: ogni layer infatti comunica con i livelli adiacenti mediante delle interfacce, cioè quell'insieme di formalismi che permettono a due livelli adiacenti di comunicare attraverso un punto di accesso (SAP). Il livello più alto comunica direttamente con l'utente, mentre quello più basso si occupa della trasmissione vera e propria dei dati. Dunque, le informazioni per essere inviate partono dal livello superiore, viaggiando per tutta la pila, e finendo poi per essere inoltrate dal livello più basso ad un altro nodo della rete. Arrivate a destinazione le informazioni ricominciano il percorso nella pila, percorrendola stavolta nel senso contrario, e arrivando finalmente all'utente destinatario.
La più evidente differenza tra i due modelli sta nel numero numero di layer: sette per il modello OSI, mentre solo cinque per il TCP/IP. In particolare il modello OSI prevede i livelli:

- *fisico*, che si occupa della codifica elettrica e della trasmissione fisica dei segnali da un nodo all'altro;
- *linea o collegamento dati*, che gestisce l'accesso al canale di comunicazione, risolvendo i problemi dovuti a collisioni e alla sincronizzazione tra mittente e ricevente;
- *rete*, che definisce l'indirizzamento e l'instradamento dei pacchetti lungo la rete;
- *trasporto*, che si occupa di assicurare ai livelli superiori una trasmissione affidabile e priva di errori;
- *sessione*, che gestisce la sincronizzazione ad alto livello dei messaggi;
- *presentazione*, che permette ad applicazioni che utilizzano standard di codifica differenti di comunicare tra loro con l'aiuto di codifiche standard;
- *applicazione*, che fornisce i servizi direttamente ai processi utilizzati dall'utente per comunicare in rete.

Nel modello TCP/IP gli ultimi tre livelli sono invece uniti all'interno di un unico strato, che viene detto applicazione. Sarà dunque compito di quest'ultimo gestire le funzioni dei livelli sessione e presentazione.
Passiamo ora dunque alla descrizione vera e propria della gestione della rete in GNU/Linux. Ricordo prima di cominciare che i primi due livelli sono sempre implementati via hardware e sono dunque quasi completamente trasparenti al sistema operativo. Essi saranno comunque descritti, ma ciò che verrà detto sarà in gran parte estensibile a tutti i sistemi operativi.

LIVELLO FISICO

Il livello fisico si preoccupa della gestione del mezzo fisico, su cui avviene la trasmissione delle informazioni, codificate in segnali elettrici binari. Partiamo subito con la loro classificazione. I mezzi trasmissivi a nostra disposizione possono essere annoverati in tre categorie principali, in funzione della tecnologia che utilizzano:

- *mezzi metallici*, che sfruttano la trasmissione di segnali elettrici, attraverso cavi rameici;
- *mezzi ottici*, che invece codificano le informazioni in radiazioni luminose, come per esempio accade all'interno dei cavi in fibra ottica;
- *mezzi wireless*, che sfruttano l'aria come canale portante per le onde elettromagnetiche, come succede ad esempio nella comunicazione fra due dispositivi Bluetooth.

Detto ciò, se vogliamo mettere a confronto mezzi di trasmissione diversi per capire quale è meglio utilizzare, dobbiamo considerare i seguenti parametri:

- *capacità del canale*, ovvero la massima quantità di informazione che il sistema riesce a trasmettere nell'unita' di tempo. L'unità di misura maggiormente utilizzata è il bit/s, assieme ai suoi multipli, Mbit/s in testa;
- *affidabilità*, cioè la probabilità che l'informazione trasmessa arrivi a destinazione priva di errori;
- *distanza massima* raggiungibile senza bisogno di ripetitori intermedi;
- *tecnologia costruttiva adottata*, appartenente alle categorie sopra esposte.

Nello stack TCP/IP non esistono specifiche riguardanti i primi due livelli: per questo motivo entrambi i livelli verranno descritti secondo lo standard per le reti locali IEEE 802.

Mezzi metallici

Il mezzo attualmente più utilizzato è, infatti, proprio quello del cavo elettrico. In particolare i più diffusi standard di cablaggio sono basati sul protocollo Ethernet IEEE 802.3 e corrispondono a:

- *10BASE-2*, che mette a disposizione, attraverso una topologia a bus, una banda di 10Mbit/s e raggiunge distanze pari a 200 m per ogni segmento. Esso utilizza un cavo coassiale di tipo RG-58, detto anche di tipo Thin Ethernet, a causa della sua maggiore sottigliezza nei confronti del cavo Thick Ethernet. Questo supporto, come gli altri coassiali, è composto da un'anima in rame, che trasporta i segnali elettrici, ricoperta da un dielettrico che isola il conduttore dalle interferenze, e a sua volta racchiuso in una maglia di rame e in una guaina esterna di plastica, tipica di tutti i cavi elettrici. Questo cavo termina con un connettore BNC a baionetta, caratterizzato da un'impedenza di 50Ω, che si innesta sulla scheda di rete e sui terminatori del bus. La codifica dei segnali è di tipo Manchester;

- *10BASE-T*, che invece utilizza una topologia fisica a stella e mette a disposizione una banda pari sempre a 10 Mbit/s. Il cavo usato non è però lo stesso: infatti vengono utilizzati cavi di categoria 3 composti da otto fili, intrecciati a due a due, di tipo schermato STP o non schermato UTP. La sostanziale differenza tra i due tipi sta però nella distanza che essi possono raggiungere: infatti mentre l'UTP può percorrere un massimo di 100 m per segmento, l'STP arriva fino a 200 m. Infine, data la topologia a stella, bisogna sottolineare come sia necessaria la presenza di un concentratore di rete, come ad esempio un hub od uno switch;

- *100BASE-T*, detta anche Fast Ethernet, è lo standard più diffuso al giorno d'oggi e differisce dalla 10BASE-T per una capacità del canale dieci volte maggiore (100 Mbit/s), per l'utilizzo di cavi di rete di categoria 5e e per la codifica dei segnali elettrici: infatti a partire da questo standard la codifica di Manchester è sostituita con la 4B/5B;
- *1000BASE-T*, detta anche Gigabit Ethernet, è lo standard che sta diffondendosi in questi tempi, e che permette di raggiungere velocità pari a 1000Mbit/s con l'utilizzo degli stessi cavi di categoria 5e e una codifica rinnovata: la 8B/10B.

Nel corso dell'anno 2007 è stato inoltre sviluppato un ulteriore standard che ancora non ha trovato però applicazione pratica: la 10 Gigabit Ethernet è infatti la più recente e veloce versione di Ethernet, disponibile sia nella versione UTP che in fibra ottica.

Mezzi wireless

Le comunicazioni wireless sono regolate dallo standard Wi-Fi IEEE 802.11: i nodi della rete sono connessi, tramite onde radio, con un access point, che permette ai PC di collegarsi alla rete wireless e di comunicare tra loro. Lo standard si divide a sua volta in:

- *802.11a*, standard caratterizzato da una velocità di 54 Mbit/s, anche se per colpa dell'intervallo di frequenze che utilizza è ormai caduto in disuso: infatti esso sfrutta una banda che sta nell'intorno dei 5 GHz, che nella maggior parte delle nazioni europee è già stata assegnata ad altri protocolli, quali ad esempio l'Hyperlan;
- *802.11b*, che presentava una velocità di soli 11 Mbit/s, sfruttando le frequenze nell'intorno dei 2,4 GHz. Anch'esso ormai non è più utilizzato da nessuno, anche se la maggior parte dei dispositivi di rete wireless lo supportano, assieme al ben più diffuso 802.11g;
- *802.11g*, senza ombra di dubbio il più diffuso al giorno d'oggi, è caratterizzato da una velocità dichiarata pari a 54 Mbit/s, anche se poi all'atto pratico questa viene circa dimezzata; esso utilizza le stesse frequenze dello standard b, e sono per questo motivo perfettamente compatibili. Inoltre è importante sottolineare come negli ultimi anni sia nato anche un ulteriore standard, chiamato Super G, che prevede una

trasmissione a due canali, al fine di raddoppiare la velocità. Purtroppo però non tutti i costruttori hanno seguito questa innovazione, consapevoli anche che questa tecnica avrebbe aumentato le interferenze elettromagnetiche sul canale;
- *802.11n*, di cui è stato rilasciato ancora solo un draft dalla IEEE, ma molti costruttori già dicono di supportarla pienamente; questa tecnologia è ormai la scommessa di molti, poiché dovrebbe portare, nella piena compatibilità con gli standard precedenti, ad un incremento della velocità pari a cinque volte: 250 Mbit/s (effettivi soli 100 Mbit/s).

Naturalmente l'adozione di questi protocolli ha il grande vantaggio di non aver bisogno di un cablaggio vero e proprio, con conseguente risparmio sia in termini di denaro che in termini di spazio. Non sono però da sottovalutare gli svantaggi: infatti una rete senza fili presenta prestazioni sempre inferiori ad una rete cablata, poiché è soggetta a maggiori interferenze e disturbi. Inoltre bisogna anche considerare i problemi di sicurezza che nascono da essa: infatti la natura broadcast di questo tipo di trasmissione implica un grosso problema di sicurezza, quale è quello delle intercettazioni di traffico (sniffing); ciò può però essere ovviato con sistemi di cifratura dei dati, quali ad esempio il WEP e il WPA.

Mezzi ottici

Le comunicazioni con mezzi ottici sono quelle che sfruttano la luce per trasmettere informazioni, e vanno quindi incluse in questo gruppo tutte le comunicazioni via laser, infrarosso o fibra ottica. Tra essi la più utilizzata è proprio quest'ultima, che funziona sfruttando le proprietà trasmissive di alcuni filamenti di vetro purificato, che guidano la luce, rendendo le informazioni estremamente immuni alle interferenze, e decodificandole attraverso l'uso di fotodiodi. La fibra ottica è utilizzata in alcuni standard IEEE 802.3 per migliorare le prestazioni dei protocolli Fast Ethernet, e sostituire il doppino telefonico in presenza di ambienti troppo inquinati dal punto di vista elettromagnetico. Le tecnologie che riguardano la fibra ottica sono:

- *100BASE-SX*, che usa due cavi, uno per ricevere e l'altro per trasmettere. È un'alternativa a basso costo alla 100BASE-FX, perché utilizza radiazioni a bassa lunghezza d'onda; ogni cavo può inoltre operare fino a distanze di 300 m e la velocità è naturalmente pari a quella di 100 Mbit/s;
- *100BASE-FX*, che utilizza la tecnologia dello standard precedente per raggiungere la stessa velocità con fibre che si possono estendere fino a 2 km di lunghezza;
- *100BASE-BX*, si differenzia dalle altre invece per l'utilizzo di un solo cavo, sia in ricezione che in trasmissione; questa necessita però di un sistema di multiplexaggio delle informazioni in arrivo;
- *1000BASE-SX*, che raggiunge una velocità di 1000 Mbit/s ed una distanza massima di 550 m, grazie alla propagazione di radiazioni luminose a bassa lunghezza d'onda;
- *1000BASE-LX*, che arriva alla stessa velocità della precedente, ma può raggiungere anche distanze di 10 km.

La fibra ottica però è ancora oggi scarsamente utilizzata rispetto al doppino telefonico per colpa del suo costo, che è spesso proibitivo, ed a causa delle continue variazioni costruttive che la interessano.

LIVELLO LINEA

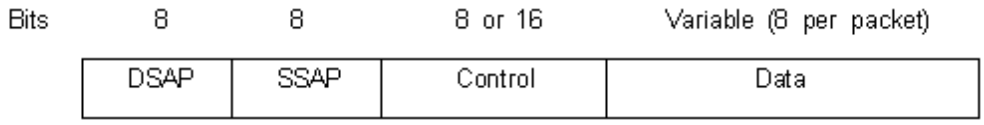
Il livello linea, o collegamento dati, viene solitamente implementato, come il livello fisico, nell'hardware di rete e nei firmware dei vari dispositivi. Questo livello è stato ugualmente standardizzato, per quanto riguarda le reti locali, nell'IEEE 802, che divide questo strato in due sottostrati con compiti diversi: LLC e MAC.

LLC

LLC (Logical Link Control) viene descritto nell'IEEE 802.2 e rappresenta il sottostrato superiore, in diretta comunicazione con il livello rete. Esso è indipendente dal mezzo di trasmissione utilizzato e dalla topologia della rete e ha come funzione principale il controllo del flusso e la gestione degli errori. Tutto ciò può essere implementato con tre tipologie differenti di servizio:

- *tipo 1*, non connesso, che permette trasmissioni unicast, multicast e broadcast, senza occuparsi della gestione delle conferme;
- *tipo 2*, connesso, che si occupa della gestione della connessione (creazione, scambio dati e rilascio), degli errori e del riordino dei pacchetti arrivati a destinazione in ordine casuale, tramite la numerazione delle trame;
- *tipo 3*, non connesso ma confermato, che si occupa solo del controllo del flusso e della gestione delle conferme. Supporta solo comunicazioni point-to-point.

La trama LLC si presenta dunque con i seguenti campi:

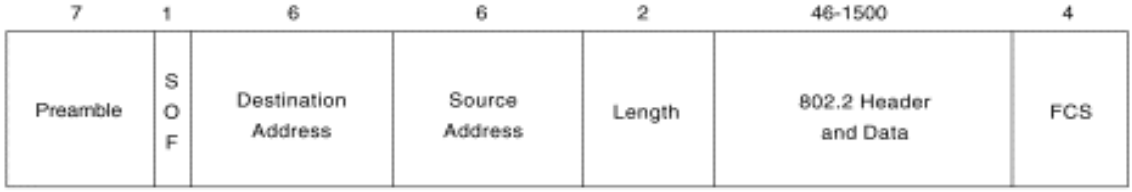


MAC

Il sottostrato MAC (Media Access Control) gestisce invece l'accesso al canale di comunicazione, divide le informazioni in più trame (framing) e provvede a sincronizzare la trasmissione di quest'ultime. Esso fornisce anche gestione degli errori a livello di bit e indirizzamento dei dispositivi di rete, tramite un indirizzo univoco a livello mondiale, detto indirizzo MAC o indirizzo fisico.
Questo sottolivello può essere gestito in vari modi, in funzione della topologia della rete e del canale adottato per la trasmissione delle informazioni. Per questo motivo esistono diversi standard all'interno dell'IEEE 802, che si possono riassumere nei tre più utilizzati e rappresentativi:

- *Ethernet v2.0 (IEEE 802.3)*: questo protocollo è dedicato alle reti configurate con topologia logica a stella o a bus, anche se quest'ultime

sono ormai deprecate visto l'elevato numero di collisioni che le caratterizzano; siamo dunque di fronte ad una rete che presenta un canale di comunicazione condiviso e che necessita l'adozione di una tecnica che ci permetta di risolvere il problema delle collisioni. Questa tecnica è detta CSMA/CD ed è basata a sua volta sulle tecniche Listen Before Talking e Listen While Talking: infatti prima di trasmettere la stazione ascolta il canale; se in questo è già presente un segnale portante essa attende un tempo casuale, per poi riprocedere al controllo del canale. Quando quest'ultimo è libero la stazione incomincia a trasmettere, rimanendo comunque in ascolto: se avverte che contemporaneamente ad essa anche un'altra stazione aveva riconosciuto il canale come libero e aveva iniziato a trasmettere, generando una collisione, allora emette un segnale di interferenza, detto jam, per far si che anche le altre stazioni si accorgano dell'accaduto; Ethernet v2.0 incapsula i dati che gli arrivano dall'LLC in trame:



I campi hanno il seguente significato:

- *Preamble*: una sequenza di segnali 1 e 0 che consente al ricevente di sincronizzare la trasmissione della trama;
- *Start Of Frame (SOF)*: anche questo campo è una sequenza di segnali 1 e 0 alternati, ma essa si conclude con due bit a livello alto e una violazione della codifica di Manchester. Questi indicano al ricevente che il prossimo bit è il primo bit vero e proprio del frame;
- *Destination Address*: il MAC address della stazione a cui è destinato il frame; quando questi bit sono posti tutti a livello alto allora la comunicazione avviene in broadcast;
- *Source Address*: il MAC address della stazione mittente;
- *Lenght*: lunghezza del campo dati a seguire;
- *Data*: sono i dati veri e propri, detti a volte anche payload. Questi propri possono ammontare al massimo a 1500 byte, ma se sono meno di 46 byte, occorre aggiungere dei byte supplementari (PAD) di riempimento per arrivare almeno a 46 byte;
- *Frame Check Sequence (FCS)*: un codice CRC calcolato sui campi precedenti seguendo lo standard CRC_32 a 32 bit.

- *Wireless Local Area Network (IEEE 802.11)*: il canale di trasmissione in questo caso è sempre condiviso ma si tratta dell'aria e la trasmissione avviene tramite antenne. A causa dei limiti insiti nelle tecnologie costruttive delle antenne sappiamo che non è possibile trasmettere e contemporaneamente rimanere in ascolto sul canale, quindi l'algoritmo proposto dal CSMA/CD non è realizzabile. Così invece di risolvere le collisioni siamo costretti a prevenirle: di questo si occupa la tecnica del CSMA/CA. Quest'ultima ha in comune con la precedente l'ascolto del canale prima di trasmettere, ma una volta trovato il canale libero non trasmette subito i dati veri e propri, ma invia un pacchetto con cui chiede all'access point o alla base station il permesso di tramettere: se questo gli viene accordato allora riceve un ACK e la trasmissione può incominciare;

- *Token Ring (IEEE 802.5)*: questo protocollo è stato sviluppato da Xerox con l'obiettivo di risolvere il problema delle collisioni, adottando una topologia logica ad anello unidirezionale, implementata però sempre con un cablaggio fisico a stella; il funzionamento si basa su un frame speciale di dimensioni ridotte, detto token, di cui ogni stazione deve entrare in possesso per acquisire il diritto di trasmettere. In situazione di inattività il token circola liberamente lungo l'anello mentre nel momento in cui il token è stato catturato da una stazione che deve trasmettere, a questo vengono allegati i dati, insieme ad altre informazioni di controllo quali per esempio indirizzo mittente e destinazione. Quando poi il token, carico, arriva al destinatario, esso viene rimesso in circolazione nella sua forma originale. Inoltre esiste una particolare stazione, detta Active Monitor, che si occupa della corretta gestione del token, proibendo la circolazione di token duplicati o loop imprevisti. Essa è solitamente la stazione con indirizzo fisico più alto, secondo la notazione esadecimale. Questa tecnica di accesso al mezzo è sicuramente molto positiva in caso di traffico elevato, in quanto risolve completamente il problema delle collisioni, ma è anche caratterizzata da un maggiore overhead, dovuto alla circolazione del token anche quando nessuna stazione vuole trasmettere.

Controller di rete

Il livello descritto in questo capitolo è implementato completamente nei controller di rete presenti nel computer. Così al sistema operativo rimane ben poco da fare, se non gestire le interfacce di rete, mettendole in comunicazione con il kernel attraverso i driver giusti. Solitamente in Unix tutti i dispositivi hardware sono rappresentati da un file (a caratteri o a blocchi, in funzione delle caratteristiche della periferica). Questo però in GNU/Linux non è vero per i controller di rete, che sono gestiti proprio dai driver, che creano le interfacce a livello software.
Per identificare le varie schede di rete, GNU/Linux usa un nome di device speciale, che non è quindi definito nella directory */dev*. Il nome utilizzato è *ethX* dove la X finale è sostituita da un numero progressivo indicante le varie schede di rete montate nel computer (eth0, eth1, ...). Solitamente per controller wireless la notazione si trasforma in *wlanX*.
Il comando che ci permette di gestire le interfacce di rete è *ifconfig*. Esso consente di individuare gli indirizzi fisici delle schede di rete, le statistiche sulle trame trasferite e le relative velocità di trasmissione, Il comando se eseguito senza opzioni né parametri visualizza le varie interfacce e i relativi dati, mentre illustreremo nel prossimo capitolo come utilizzare *ifconfig* per la configurazione del livello rete.
Infine per quanto riguarda le caratteristiche specifiche dei controller wireless, possiamo individuarle con il comando *iwconfig*. Come il precedente se eseguito senza opzioni esso ci permette di visualizzare i dati sul collegamento: la qualità del segnale, l'indirizzo fisico, l'ESSID dell'access point associato, le modalità di trasmissione e la frequenza utilizzata per la comunicazione wireless. Se invece non si è associati a nessun access point possiamo cercare reti wireless con il comando:

iwlist scan <interfaccia wireless>

Una volta individuato l'ESSID della rete a cui ci vogliamo connettere possiamo eseguire, con permessi di root, il comando:

iwconfig <interfaccia> essid <ESSID della rete> key <chiaveWEP>

Ovviamente se la rete è aperta non è necessario specificare l'opzione key, mentre se è protetta con il protocollo WPA è necessario un ulteriore software che proceda con l'autenticazione: WPA Supplicant.

LIVELLO RETE

Il livello rete è il primo livello, percorrendo lo stack verso l'alto, che viene implementato via software dal kernel del sistema operativo: in particolare in Linux questo è solitamente implementato in diversi moduli, che solitamente sono compilati direttamente all'interno del kernel monolitico.
La funzione principale di questo livello in una rete TCP/IP è quella di implementare il cosiddetto Internetworking, ovvero permettere lo scambio di informazioni tra calcolatori appartenenti a sottoreti diverse, anche in caso di incompatibilità o differenze nei domini attraversati. Ciò viene realizzato attraverso:

- *indirizzamento*, ovvero l'assegnamento di un numero univoco in tutta la rete, che distingua una stazione (nodo) dalle altre;
- *instradamento*, cioè quella funzione per cui un nodo commutatore (router) all'arrivo di un pacchetto da una sottorete, sceglie il percorso migliore che questo dovrà fare per arrivare a destinazione, consultando un apposita tabella (routing table) e spedendo il pacchetto in un'altra sottorete; la routing table è aggiornata da protocolli appositi che utilizzano algoritmi specifici.

Nel modello OSI erano previste altre funzioni, quali il controllo della congestione e la garanzia della qualità del servizio, ma queste non sono implementate nei protocolli che andremo a descrivere. In particolare i protocolli che ci interessano sono classificabili in tre tipi differenti:

- *instradati*, che incapsulano i dati in pacchetti, che contengono anche importanti informazioni di controllo, quali ad esempio l'indirizzo mittente e quello destinatario;
- *di instradamento*, cioè quei protocolli che servono ai nodi commutatori della rete per compilare ed aggiornare le proprie tabelle di routing, al fine di poter stabilire in ogni momento il percorso migliore, in base a parametri quali costo, tempo di percorrenza e distanza fisica. Questi parametri vengono detti metriche;
- *di controllo*, ovvero protocolli che si occupano di tramettere messaggi di controllo o di errore, oppure di reperire informazioni dai nodi presenti nella rete.

Procediamo ora con l'analizzare le principali implementazioni dei protocolli descritti sopra.

IP

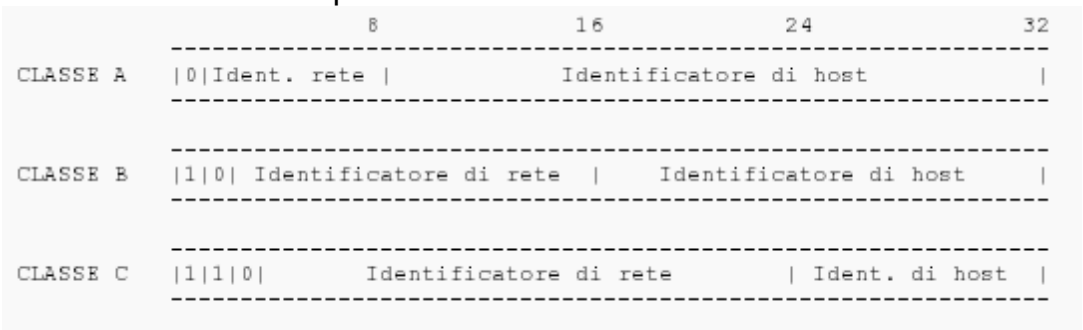
Il protocollo instradato utilizzato nello stack TCP/IP è l'IP (Internet Protocol). IP è un protocollo che lavora in modalità non connessa non affidabile e che utilizza

come tecnica di commutazione quella a pacchetto: infatti esso divide le informazioni in pacchetti, a cui associa un intestazione, e li trasmette. I pacchetti IP vengono anche detti datagrammi, poiché non c'è nessuna garanzia di arrivo. Questi però possono giungere a destinazione in ordine sparso, utilizzando magari percorsi differenti, oppure possono anche andare persi o arrivare danneggiati: sarà poi compito del livello superiore controllare ognuna di queste eventualità. Il pacchetto IP si compone nel seguente modo:

+	Bits 0–3	4–7	8–15	16–18	19–31
0	Version	Internet Header length	Type of Service (now DiffServ and ECN)	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options (optional)				
160 o 192+	Data				

- *Versione*: indica la versione del pacchetto IP; esistono infatti due versioni al momento di questo protocollo: la versione da noi descritta è la 4 (per questo motivo esso è detto anche IPv4);
- *Internet Header Length*: indica la lunghezza dell'header del pacchetto IP; tale lunghezza può variare da 20 a 60 byte, in funzione della presenza e della lunghezza del campo facoltativo Options;
- *Type of Service*: nelle specifiche iniziali questo campo serviva all'host mittente per specificare il modo con cui l'host ricevente doveva trattare il pacchetto; recentemente questi 8 bit sono stati ridefiniti e ora sono utilizzati per le applicazioni di streaming in real-time;
- *Total Length*: indica la dimensione dell'intero pacchetto; tale lunghezza può variare da un minimo di 20 byte (header minimo e campo dati vuoto) ad un massimo di 65535 byte;
- *Identification*: utilizzato per identificare univocamente i vari frammenti in cui può essere diviso un pacchetto IP;
- *Flags*: campo utilizzato per il controllo del protocollo e della frammentazione dei datagrammi;
- *Fragment Offset*: indica l'offset di un particolare frammento relativamente all'inizio del pacchetto IP originale;
- *Time To Live (TTL)*: indica il numero massimo di hop, ovvero di salti da nodo a nodo, che il pacchetto può ancora fare, al fine di evitarne la persistenza indefinita sulla rete: ogni router che riceve il pacchetto prima di inoltrarlo ne decrementa il TTL, quando questo arriva a zero il datagramma non viene più inoltrato ma scartato e viene mandato al mittente un messaggio di errore ICMP;
- *Protocol*: indica il protocollo che viene incapsulato nel campo dati;
- *Header Checksum*: è un campo usato per il controllo degli errori nell'header, utilizzando la tecnica del checksum; da notare come non vi siano effettivi controlli sul campo dati;
- *Source address*: indica l'indirizzo dell'host mittente;
- *Destination address*: indica l'indirizzo dell'host destinatario;
- *Options*: campo facoltativo che contiene le opzioni per usi più specifici del protocollo.

I campi più interessanti del pacchetto IP però sono gli indirizzi, ognuno dei quali è sempre formato da 32 bit e può essere diviso in quattro gruppi di 8 bit ciascuno, espressi come numeri da 0 a 255.
Ci troviamo però di fronte al problema di suddividere la rete TCP/IP (Internet) in sottoreti di dimensione arbitraria, ma caratterizzate da indirizzi simili. Inizialmente sono per questo motivo state individuate cinque classi di indirizzi, ma per i fini della nostra esposizione ne descriveremo solo tre:



In questa classificazione i tipi di indirizzi sono composti da tre parti fondamentali: una parte che rimane sempre costante, una che identifica la sottorete e l'ultima che identifica univocamente l'host. Oggi però si è notato come questa classificazione sia troppo rigida, così si è passati al CIDR (Classless Inter-Domain Routing) che permette di definire indirizzi e sottoreti di appartenenza secondo la notazione:

a.b.c.d/x

dove x è il numero di bit contigui (a partire dal primo a sinistra) che individua la sottorete di appartenenza dell'host.
Esiste inoltre un altro tipo di classificazione per quanto riguarda gli indirizzi:

- *indirizzi pubblici*, ovvero quegli indirizzi che individuano host affacciati direttamente sulla rete Internet e che vengono assegnati dalle autorità competenti agli ISP e che a loro volta li distribuiscono, in modo dinamico o statico, agli utenti della rete;
- *indirizzi privati*, che vengono usati dagli host di reti LAN private, non connesse ad Internet, oppure connesse attraverso un sistema di traduzione degli indirizzi privati in indirizzi pubblici (NAT); gli indirizzi riservati ad uso privato sono:
  - *10.0.0.0/8*: 10.0.0.0 → 10.255.255.255;
  - *172.16.0.0/12*: 172.16.0.0 → 172.31.255.255;
  - *192.168.0.0/16*: 192.168.0.0 → 192.168.255.255;

Affrontiamo ora un problema pratico: vogliamo configurare una rete privata TCP/IP con host GNU/Linux e senza sottoreti.
I parametri del livello rete da configurare sono i seguenti:

- *indirizzo IP*, completo di tutti e quattro i byte;
- *subnet mask*, ovvero un parametro che è formalmente identico ad un indirizzo IP, ma che ha la caratteristica di possedere a livello alto solo i bit che nell'indirizzo rappresentano la sottorete di appartenenza. Quindi se calcoliamo l'and logico bit a bit tra indirizzo e maschera otteniamo un nuovo indirizzo, che viene detto indirizzo di rete e che si differenzierà dall'indirizzo IP reale per il valore assunto dai bit che individuano l'host, sempre e comunque a livello basso;

Per configurare questi due parametri il comando di shell è il solito *ifconfig*, nella sintassi:

ifconfig <interfaccia> <indirizzo host> netmask <subnet mask>

Configurando quindi con questo comando tutti gli host possiamo permettere

loro di comunicare, almeno per quanto riguarda il livello rete. Naturalmente tutti gli host devono condividere la stessa subnet mask e lo stesso indirizzo di rete, in modo tale che essi differiscano solo per i valori dei bit specifici di ogni host.

DHCP

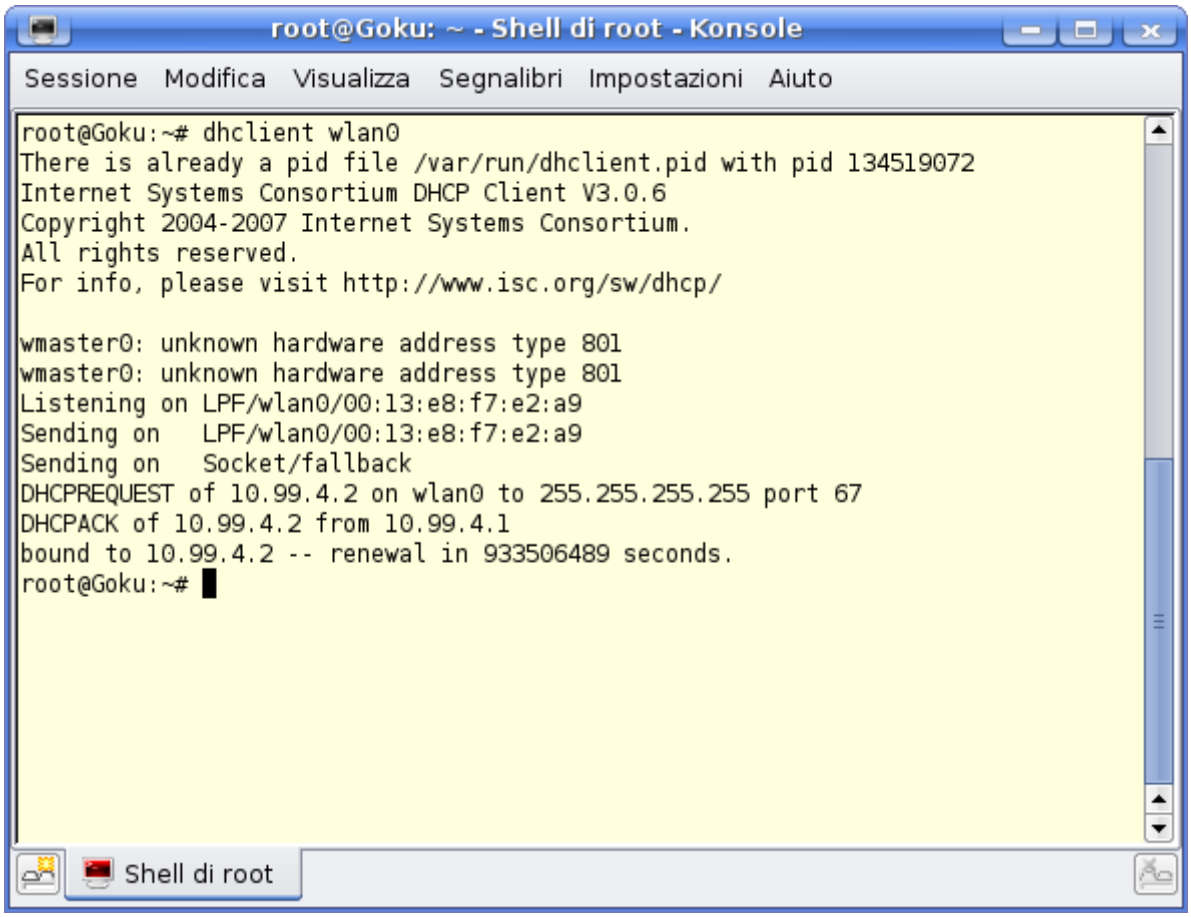
Esiste inoltre anche un metodo dinamico ed automatico per la configurazione degli indirizzi IP e delle relative subnet mask. Questo metodo è implementato nel protocollo DHCP (Dynamic Host Configuration Protocol), che, anche se appartiene al livello applicazione, verrà qui descritto. Se è presente nella stessa sottorete un server DHCP questo è in grado di assegnare ad ogni host che lo chiede, detto client DHCP, un indirizzo IP, la maschera di sottorete e, come vedremo meglio poi, anche default gateway e gli indirizzi dei server DNS. L'assegnamento si compone di quattro fasi principali:

- *DCHP Discovery*, in cui il client invia in broadcast un pacchetto, in cerca di un server DHCP che lo intercetti. In questa fase l'indirizzo IP del mittente è 0.0.0.0 mentre quello del destinatario è 255.255.255.255, ad indicare una trasmissione broadcast;
- *DHCP Offer*, in cui i server DHCP presenti nella rete riservano per il client uno degli indirizzi disponibili e glielo offrono, assieme alla maschera di sottorete, in un pacchetto; in questa fase quindi l'indirizzo del mittente è definito mentre la destinazione sarà sempre 255.255.255.255, poiché ancora il destinatario non ha un indirizzo assegnato;
- *DHCP Request*, in cui il client, dopo aver ricevuto le offerte di assegnazione, sceglie quale offerta accettare e manda un pacchetto di richiesta, sempre in broadcast, contenente l'indirizzo del server che ha scelto e l'indirizzo IP che gli era stato offerto. In questo modo i server che non sono stati scelti possono rilasciare l'indirizzo che avevano riservato, mentre il server giusto può procedere con la quarta fase; l'intestazione IP anche stavolta è caratterizzata da un indirizzo sorgente 0.0.0.0 e da un destinatario pari a 255.255.255.255;
- *DHCP Pack*, in cui il server invia, sempre in broadcast, un pacchetto contente tutte le informazioni necessarie di cui il client necessita per accedere alla rete; similmente alla seconda fase solo l'indirizzo sorgente è precisato, mentre il destinatario è l'indirizzo broadcast 255.255.255.255.

DHCP offre la grande comodità di una gestione centralizzata della rete, che consente all'amministratore di non dover intervenire singolarmente su ogni PC quando vuole cambiare dei parametri di configurazione. Basta infatti agire solo sul server DHCP, poiché i client poi, al rinnovo dell'indirizzo, aggiorneranno la loro configurazione.
In GNU/Linux per configurare un macchina in DHCP basta dare il seguente comando da root:

dhclient <interfaccia>

e il nostro computer si trasformerà in un client DHCP alla ricerca di un server che possa assegnargli un indirizzo.



Per quanto riguarda invece il server DHCP questo ruolo è solitamente svolto dai router, che lo integrano al loro interno, ma può anche essere svolto da un host della rete. Una delle implementazioni che troviamo in GNU/Linux di server DHCP è rappresentata da un demone, chiamato *Udhcpd*, che appena installato si autoconfigura per l'esecuzione automatica all'avvio del sistema. La configurazione del server avviene editando il file */etc/udhcpd.conf* nel seguente modo:

start <primo indirizzo assegnabile>
end <ultimo indirizzo assegnabile>
max_leases <massimo numero di indirizzi assegnabili allo stesso tempo>
lease_file <percorso file dove sono memorizzati gli indirizzi assegnati>
option dns <indirizzo server dns>
option subnet <subnet mask>
option router <indirizzo default gateway>
option lease <secondi per cui un indirizzo può rimanere assegnato>

Routing

Ogni rete, come abbiamo detto, può essere composta da sottoreti diverse. Queste per comunicare tra loro devono essere collegate insieme da un router. Ogni router potrà poi comunicare con ogni sottorete a cui è connesso attraverso l'interfaccia a cui essa è collegata. Ogni interfaccia del router si comporterà come un vero e proprio host, con la differenza che essa potrà essere vista da ogni host della sottorete come porta di accesso (gateway) alle altre sottoreti. Poiché un router possiede più interfacce deve contenere al suo interno una speciale tabella, detta tabella di routing, che conterrà informazioni

sull'interfaccia in cui inoltrare i pacchetti, in funzione dell'indirizzo di rete di destinazione. Inoltre questa tabella sarà presente anche in ogni host, e servirà a determinare quale host contattare per inviare pacchetti. Naturalmente nel caso destinatario e mittente siano sulla stessa rete non sarà necessario contattare un router, ma la trasmissione può essere diretta. Importante è però il problema dell'aggiornamento delle tabelle di routing, poiché la morfologia della rete può variare nel tempo, sia a causa di scelte umane sia per colpa di guasti. Questo però non deve risultare mai un problema poiché la rete deve saper gestire questi eventi, adattandosi e riaggiornando ogni volta le tabelle di routing. Esistono due modi fondamentali di aggiornare le tabelle:

- *statico*, che consiste nell'aggiornamento manuale delle voci della tabella in ogni host e router; questo metodo è poco scalabile, poiché non si sa adattare da solo agli eventi che possono occorre durante il tempo;
- *dinamico*, attraverso cui le tabelle vengono popolate dagli algoritmi di routing, che permettono ai router di scambiarsi informazioni sulla rete e quindi di ricalcolare ogni volta i percorsi possibili per le destinazioni conosciute.

Riprendiamo l'esempio fatto in precedenza. Dopo aver assegnato al nostro host GNU/Linux un indirizzo IP e una subnet mask, decidiamo di andare ad aggiungere informazioni alla sua tabella di routing, per permettergli di raggiungere anche host di altre sottoreti. Ipotizziamo dunque che l'indirizzo dell'host sia 192.168.1.3 e quello dell'interfaccia del router collegata alla sottorete 192.168.1.0/24 sia 192.168.1.1. Sarà dunque questo il nostro gateway. Il comando che ci permette di andare a modificare la tabella di routing è *route*. Il comando senza opzioni stampa a video la tabella mentre la sintassi per modificarla è la seguente:

route [add|del] <indirizzo di rete destinazione> gw <indirizzo gateway>

L'opzione principale è la scelta fra *add* e *del* che ci permette di aggiungere o rimuovere voci alla tabella. In particolare, quando vogliamo contattare lo stesso gateway per ogni sottorete di destinazione, allora esso prende il nome di default gateway e l'indirizzo di rete destinazione è sostituito dalla parola *default*. Da ricordare che l'opzione *default* è valida per tutte le destinazioni non specificate nelle altre voci della tabella e che ogni interfaccia dell'host può avere un suo default gateway e delle regole di instradamento precise.

ICMP

Il protocollo ICMP invece fa parte della categoria dei protocolli di controllo. Anche se questo protocollo appartiene al livello rete, esso viene incapsulato in IP per poi essere passato ai livelli sottostanti. Il pacchetto ICMP parte dunque dal bit 160 del datagramma IP:

	Bits 160-167	168-175	176-183	184-191
160	Type	Code	Checksum	
192	ID		Sequence	

- *Type* e *Code* servono ad identificare di che tipo di messaggio ICMP si tratta. I più importanti e che analizzeremo in questa tesina sono:
  - *Echo Reply*;

- *Destination Unreachable*;
- *Echo Request*;
- *Time Exceeded*;

- *Checksum* rappresenta il solito controllo sull'integrità del pachetto;
- *ID* e *Sequence* sono due campi ideati per identificare i pacchetti di tipo *Echo Reply*;

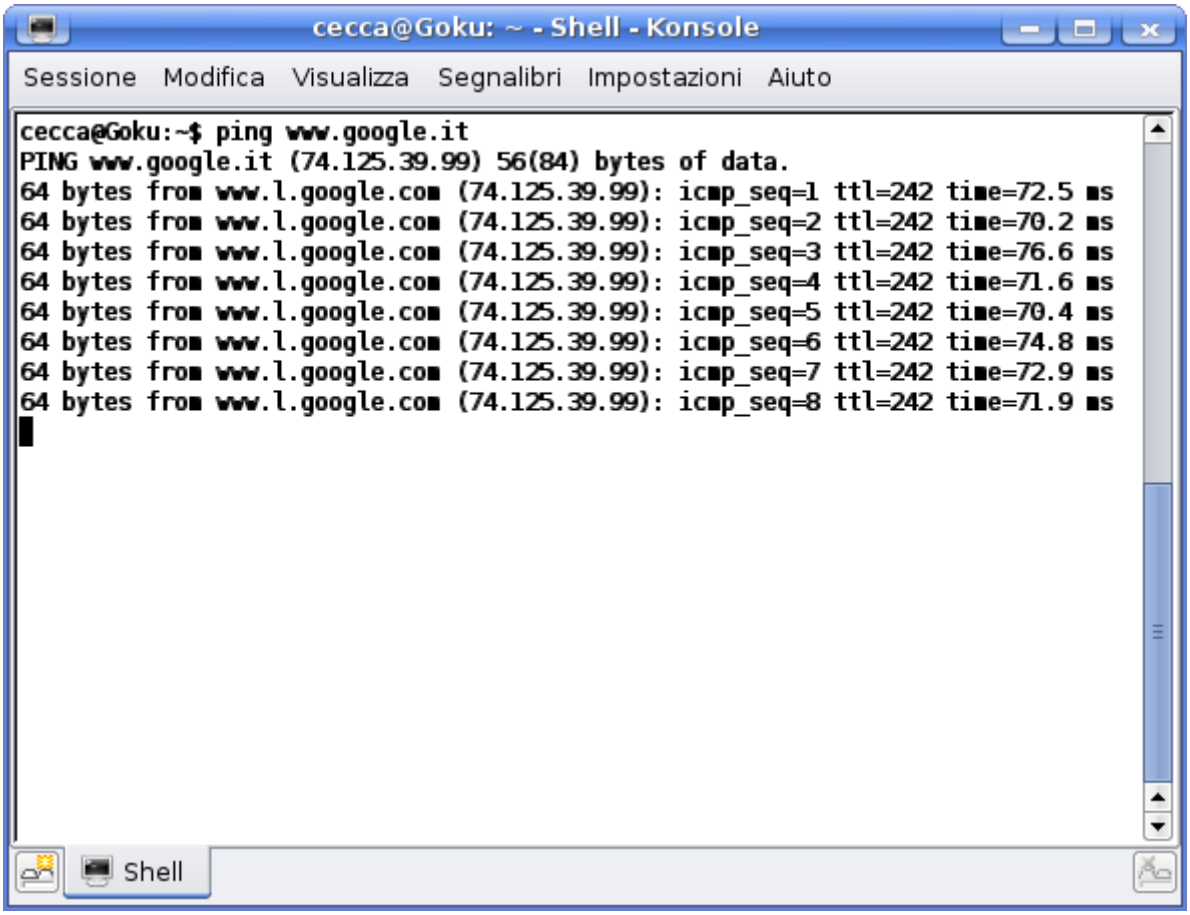
Il protocollo ICMP trova le sue più importanti applicazioni nei comandi *ping* e *traceroute*.
Il comando *ping* è usato per rilevare la presenza o la raggiungibilità di un computer della rete. Esso è composto da due fasi principali:

- il computer che decide di effettuare il ping manda un *Echo Request* all'host destinazione, ovvero una richiesta di risposta per capire se esso può essere raggiunto;
- il computer destinazione, quando riceve l'*Echo Request* risponde con un *Echo Reply*, per far capire al mittente che ha ricevuto il suo pacchetto. Se anche il mittente è raggiunto dal pacchetto ICMP allora i computer possono comunicare tranquillamente;

Ogni distribuzione possiede il comando ping; esso invia *Echo Reply* senza mai smettere e viene interrotto solo quando l'utente decide di farlo. Ecco la sintassi:

ping <opzioni> <indirizzo destinazione>

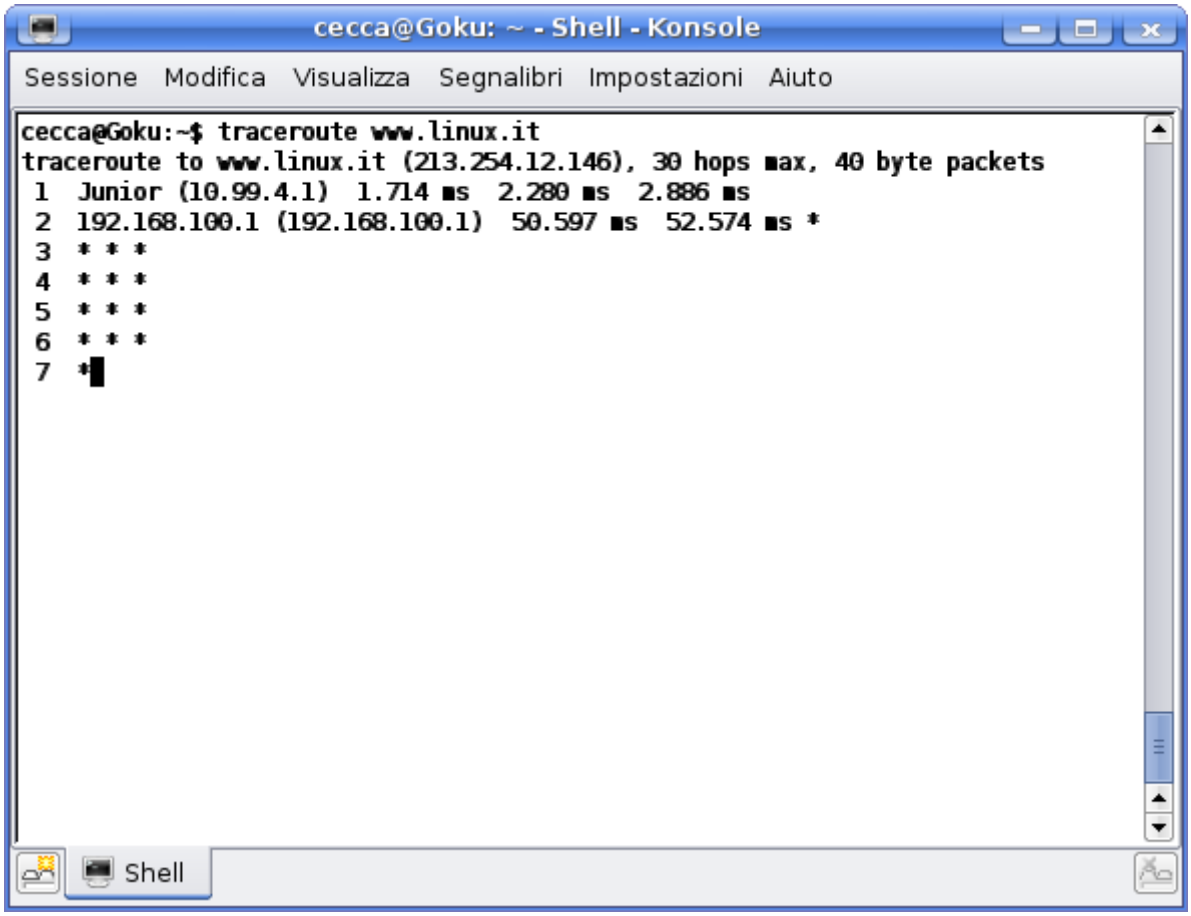
dove l'opzione più usata è *-s* con cui possiamo decidere arbitrariamente la dimensione dei pacchetti da inviare.



Il comando *traceroute* è invece utilizzato per determinare il percorso che i pacchetti intraprendono per arrivare ad una specificata destinazione; ciò è implementato sfruttando il meccanismo del TTL, per cui ogni router che scarta un pacchetto con TTL uguale a zero deve mandare al mittente un pacchetto di

errore ICMP di tipo *Time Exceeded*. Se inviamo dunque pacchetti verso un host destinazione con TTL prima uguale a uno, poi uguale a due e così via, il datagramma che avrà TTL uguale a uno verrà scartato dopo un solo hop, il datagramma con TTL uguale a due verrà scartato dopo due hop e così via. All'arrivo di ogni messaggio di errore verrà poi estratto l'indirizzo mittente che ci permetterà di ricavare il percorso tra noi e l'host destinazione. Questo comando in GNU/Linux ha la seguente sintassi:

traceroute <indirizzo destinazione>



ARP

Un altro problema che non abbiamo ancora trattato nasce quando il datagramma IP viene passato al livello datalink, in particolare quando prima della trasmissione vera e propria abbiamo la necessità in una rete locale di capire quale sia l'indirizzo fisico del destinatario. Infatti nel caso il destinatario sia sulla stessa sottorete il livello linea deve incapsulare il pacchetto IP in un frame con l'indirizzo fisico del destinatario, mentre se i due computer non appartengono alla stessa sottorete il mittente deve inserire nel frame Ethernet il MAC address del gateway. Abbiamo dunque bisogno di una mantenere una corrispondenza tra indirizzi a livello rete ed indirizzi a livello linea, per quanto riguarda gli host della nostra sottorete. Per far ciò sfruttiamo le potenzialità del protocollo ARP (Address Resolution Protocol): esso infatti mantiene in ogni nodo in cui è implementato una cache in cui memorizza la coppia di indirizzi, fisico e IP, per ogni host recentemente contattato. Quindi quando abbiamo bisogno di risalire all'indirizzo fisico da quello IP il protocollo consulta la sua memoria, che viene quindi detta cache ARP. Se però non è presente l'indirizzo che cerchiamo

allora il protocollo procede nelle seguenti fasi:

- *ARP Request*, in cui viene inviata una richiesta in broadcast che contiene l'indirizzo IP che si vuole mappare, l'IP mittente e il MAC address mittente;
- *ARP Reply*, che parte quando l'host destinatario riceve l'*ARP Request* e, quindi, consiste nell'inviare un ulteriore pacchetto di risposta al mittente dove viene indicato il MAC address del destinatario;

Per svolgere queste funzioni il protocollo ARP non deve ricorrere come l'ICMP all'incapsulamento in IP, ma possiede un pacchetto tutto suo:

+	Bits 0 - 7	8 - 15	16 - 31
0	Hardware type (HTYPE)		Protocol type (PTYPE)
32	Hardware length (HLEN)	Protocol length (PLEN)	Operation (OPER)
64	Sender hardware address (SHA)		
?	Sender protocol address (SPA)		
?	Target hardware address (THA)		
?	Target protocol address (TPA)		

- *Hardware Type (HTYPE)*, dove viene indicato il tipo di protocollo di livello linea, codificato attraverso un numero identificativo;
- *Protocol Type (PTYPE)*, dove invece abbiamo l'identificativo del protocollo di livello rete;
- *Hardware Length (HLEN)*, dove abbiamo la lunghezza espressa in byte dell'indirizzo fisico;
- *Protocol Length (PLEN)*, in cui è specificata la lunghezza dell'indirizzo di livello rete;
- *Operation*, che assume valore 1 se il pacchetto è un *ARP Request* mentre assume valore 2 se è un *ARP Reply*;
- *Sender hardware address (SHA)*, che contiene l'indirizzo fisico del mittente;
- *Sender Protocol Address (SPA)*, che contiene l'indirizzo di livello rete, sempre del mittente;
- *Target Hardware Address (THA)*, in cui troviamo l'indirizzo fisico del destinatario; questo campo viene lasciato vuoto nel caso di *ARP Request*;
- *Target Protocol Address (TPA)*, dove abbiamo l'indirizzo di livello rete del destinatario del pacchetto.

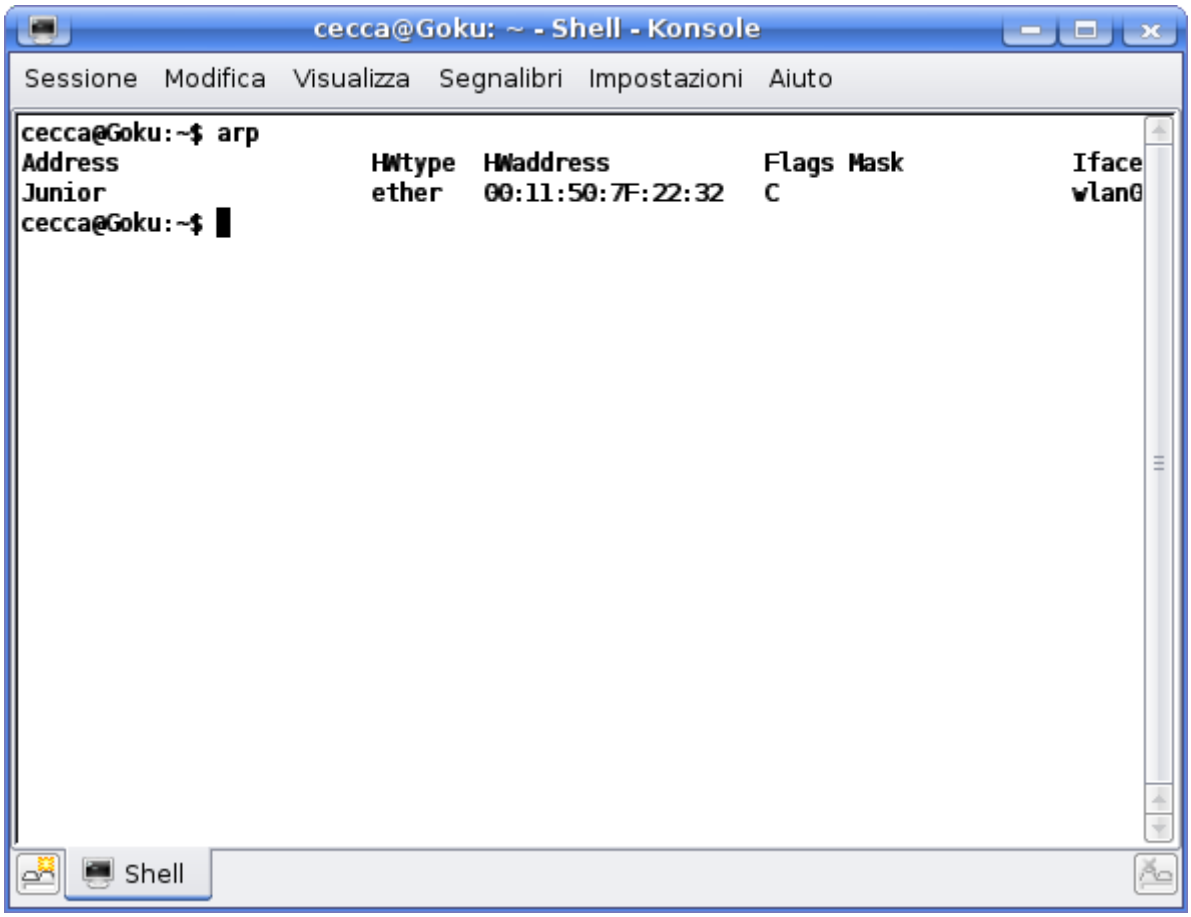
In GNU/Linux la cache ARP si può consultare e, avendo i permessi di root, anche editare, aggiungendo e rimuovendo voci. Il comando da shell è il seguente:

arp <opzioni>

dove le opzioni più usate sono:

- *-d <indirizzo IP>*, che cancella la voce corrispondente all'indirizzo fornito;
- *-s <indirizzo IP> <MAC address>*, che aggiunge una voce alla cache;

Infine senza opzioni aggiuntive il comando stampa a video il contenuto della cache, organizzato in una tabella che ha come campi l'indirizzo IP, il protocollo usato a livello linea, l'indirizzo fisico, l'interfaccia del PC su cui è raggiungibile l'host e alcuni flag di controllo.



LIVELLO TRASPORTO

Il livello trasporto è il primo livello in cui non ci occupiamo minimamente della gestione della sottorete di comunicazione: quindi tutti i problemi di trasmissione, commutazione e calcolo del percorso sono completamente ignorati, in quanto la comunicazione si ipotizza sia end-to-end, come nelle connessioni punto punto. Questo livello è però anche l'ultimo livello orientato al fornitore, dunque nel livello applicazione non sentiremo più parlare di controllo della congestione, controllo del flusso o rilevazione dell'errore. Infatti dopo questo layer tutti questi problemi devono essere già stati completamente risolti, in modo tale da fornire al livello soprastante una visione della rete affidabile e sicura. In poche parole l'obiettivo principale di questo livello è fornire alle applicazioni delle primitive che consentano agli sviluppatori di non preoccuparsi di come è implementata la rete sottostante, ma di pensare solo ed esclusivamente a risolvere i problemi che riguardano i loro applicativi di rete.
Per far ciò questo livello mette a disposizione i seguenti servizi, che per certi versi lo rendono simile al livello datalink:

- *creazione della connessione*, *trasferimento dei dati* e *rilascio* di essa, che ci permettono di avere un servizio connesso;
- *controllo del flusso*, ovvero quel servizio che si preoccupa di far arrivare i pacchetti in numero tale da non mandare mai in overflow i buffer del destinatario;
- *controllo della congestione*, cioè la gestione di quelle situazioni in cui alcuni nodi della rete sono intasati dall'eccessivo traffico;

- *controllo degli errori*, che assicura al mittente che i pacchetti arrivino a destinazione privi di errori; in particolare ciò viene implementato con l'invio di un acknowledgment (conferme) da parte del destinatario. Queste saranno positive se i pacchetti sono arrivati in modo corretto, altrimenti saranno negative e saranno susseguite dalla ritrasmissione dei dati; se non dovesse arrivare la conferma al mittente entro un tempo limite esso procede comunque alla ritrasmissione;
- *arrivo ordinato dei pacchetti*, su cui il livello rete non dà nessuna garanzia;
- *indirizzamento*, che consiste nell'assegnare un T_SAP ad ogni applicazione che sta utilizzando il livello di trasporto;
- *multiplexing*, ovvero la capacità di avere più connessioni a livello trasporto su un'unica entità di livello rete (multiplexing upward) oppure di avere più connessioni a livello rete per un'unica entità a livello trasporto (multiplexing downward). Il primo servizio ci dà un vantaggio in termini di costi, poiché possiamo avere su ogni host più connessioni ad applicazioni diverse, mentre il secondo ci dà un vantaggio in termini di banda, poiché il traffico di una stessa applicazione viene ripartito su più linee fisiche.

I protocolli più usati nel livello trasporto delle reti TCP/IP sono due: UDP e TCP.

Porte e socket

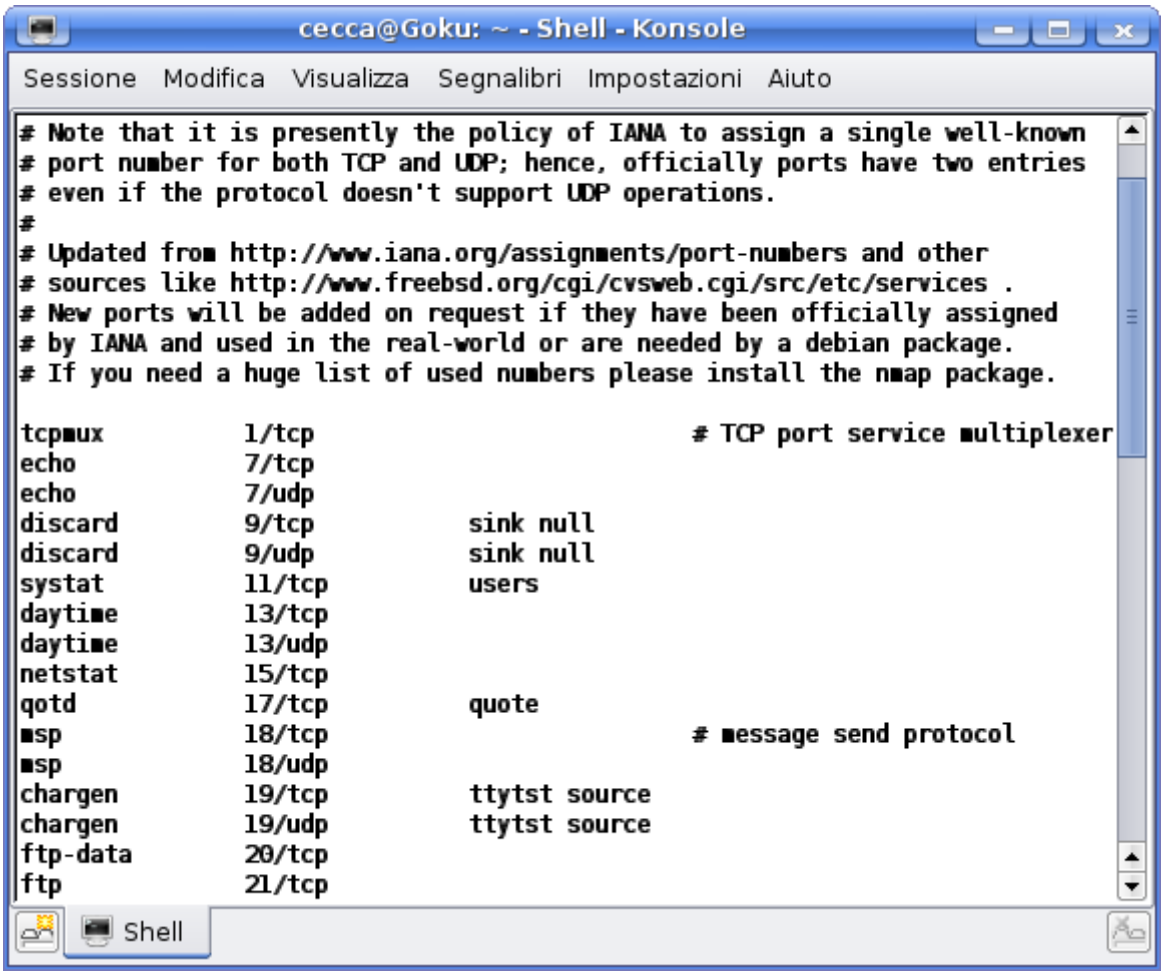
Multiplexing e indirizzamento sono entrambi gestiti con il sistema delle porte, che nella terminologia usata dal modello ISO/OSI, corrispondono ai T_SAP. Su ogni indirizzo IP vengono dunque create 65535 porte (multiplexing downward), ognuna delle quali può essere usata da un processo server per porsi in attesa di una richiesta di connessione da parte di un client; quindi per contattare univocamente un processo in ascolto abbiamo bisogno della tripla di parametri:

- *indirizzo IP*;
- *numero di porta*;
- *protocollo di trasporto usato*.

Questa tripla è detta socket o end-point ed ogni connessione è identificata univocamente a sua volta da due end-point.
Le porte sono divise in gruppi, in funzione del loro utilizzo:

- *le prime 1024* sono dette Well Known Ports e sono state assegnate dalle organizzazioni competenti ognuna ad un particolare servizio; questo invitta ogni server che fornisce quel servizio ad essere in ascolto su una particolare porta, in modo tale che i client che lo vogliano contattare sappiano sempre dove trovarlo;
- *le porte dalla 1024 alla 49151* sono invece assegnate, sempre staticamente, dalle organizzazioni alle ditte che ne fanno richiesta per i loro programmi di rete;
- *tutte le rimanenti* invece sono assegnate dal sistema operativo, in modo dinamico, quando un client apre una connessione. Queste infatti non hanno la necessità di comporre un socket prettamente identificabile, poiché solo i server offrono servizi e hanno la necessità di essere contattati su end-point noti a priori.

In GNU/Linux la corrispondenza tra socket e servizi assegnati è mantenuta nel file di testo */etc/services*. A seguire parte del file:



UDP

Il protocollo UDP (User Datagram Protocol) è un protocollo di livello trasporto che fornisce servizi non connessi non affidabili, perciò non si preoccupa minimamente di gestire le connessioni, controllare il flusso o le congestioni, oppure di spedire conferme, positive o negative, al mittente. Gli unici servizi che fornisce sono quelli dell'indirizzamento, del multiplexing upward e di una rilevazione degli errori, tra l'altro opzionale, che porta, in caso di datagramma corrotto, allo scarto dello stesso senza l'invio di nessun acknowledgment negativo al mittente. Questo protocollo risulta dunque inaffidabile, ma molto veloce a causa del bassissimo overhead: molto efficiente dunque si dimostra nella trasmissione in real-time di informazioni, come ad esempio nelle applicazioni di streaming video/audio o nelle chiamate VoIP. Il datagramma UDP è quindi composto da pochi campi:

+	Bit 0-15	16-31
0	Source Port (optional)	Destination Port
32	Length	Checksum (optional)
64+	Data	

- *Source Port*, che identifica la porta sul computer mittente;

- *Destination Port* , che identifica invece il numero di porta del destinatario;
- *Length*, che contiene la lunghezza totale del datagramma;
- *Checksum*, che contiene il codice di rilevazione degli errori presenti nel datagramma, secondo l'algoritmo appunto del checksum;
- *Data*, che rappresenta l'insieme dei dati da trasportare.

Infine, UDP viene anche utilizzato per applicazioni multicast e broadcast.

TCP

Il TCP (Trasmission Control Protocol) è un protocollo orientato al flusso di byte bidirezionale (full-duplex) che fornisce alle applicazioni servizi connessi e affidabili, completi di controllo del flusso, controllo delle congestioni e tutti gli altri servizi che prima abbiamo elencato come caratteristici di questo livello. In particolare:

- le connessioni vengono create con l'utilizzo della tecnica tree-way-handshake: il client invia una richiesta di connessione al server, questo risponde con una conferma e infine il client invia un ulteriore acknowledgment, assieme ai primi dati; questa tecnica è importante per evitare l'apertura involontaria di connessioni causata da pacchetti duplicati che girano per la rete;
- le connessioni vengono chiuse in modalità simmetrica: ognuno dei due computer può chiudere la propria connessione, ma continuare a ricevere finché l'altro non chiude la propria;
- il flusso di byte viene frazionato in segmenti, che possono anche arrivare a destinazione in disordine, ma grazie ad uno speciale numero di sequenza questi possono essere riordinati;
- il controllo del flusso è implementato con la tecnica sliding-window, che consiste nel mantenere in ogni computer un buffer (che sarà di invio o di ricezione), la cui dimensione è detta finestra; questa non ha un valore fisso, ma è variabile, poiché in ricezione essa sarà data dal numero dell'ultimo byte che essa può ricevere meno l'ultimo byte confermato, mentre in trasmissione sarà uguale al numero dell'ultimo byte pronto ad essere trasmesso meno l'ultimo byte di cui è già arrivata conferma;
- il controllo della congestione è anch'esso implementato con la tecnica della finestra, che in questo caso viene detta finestra di congestione e rappresenta il massimo numero di byte che la rete è in grado di trasmettere in quel momento; il numero effettivo di byte che il mittente può trasmettere è dunque dato dal valore minimo tra finestra di ricezione del destinatario e finestra di congestione;
- gli acknowledgment sono inviati con la tecnica del piggybacking, che permette di ridurre il traffico di rete includendo le conferme negli stessi segmenti dati. Quando però un acknowledgment non arriva al mittente entro un certo intervallo di tempo, esso procede alla ritrasmissione del segmento; da ciò nasce il rischio della duplicazione dei segmenti, poiché le conferme possono anche subire ritardi a causa di congestioni incontrate lungo la rete.

Tutti questi servizi rendono il TCP un protocollo molto più lento rispetto all'UDP, anche se la maggior parte dei servizi utilizza il TCP, visto che esso offre un'alta garanzia di affidabilità e robustezza

I servizi già elencati vengono implementati attraverso variabili che troviamo nei campi del segmento TCP; in particolare:

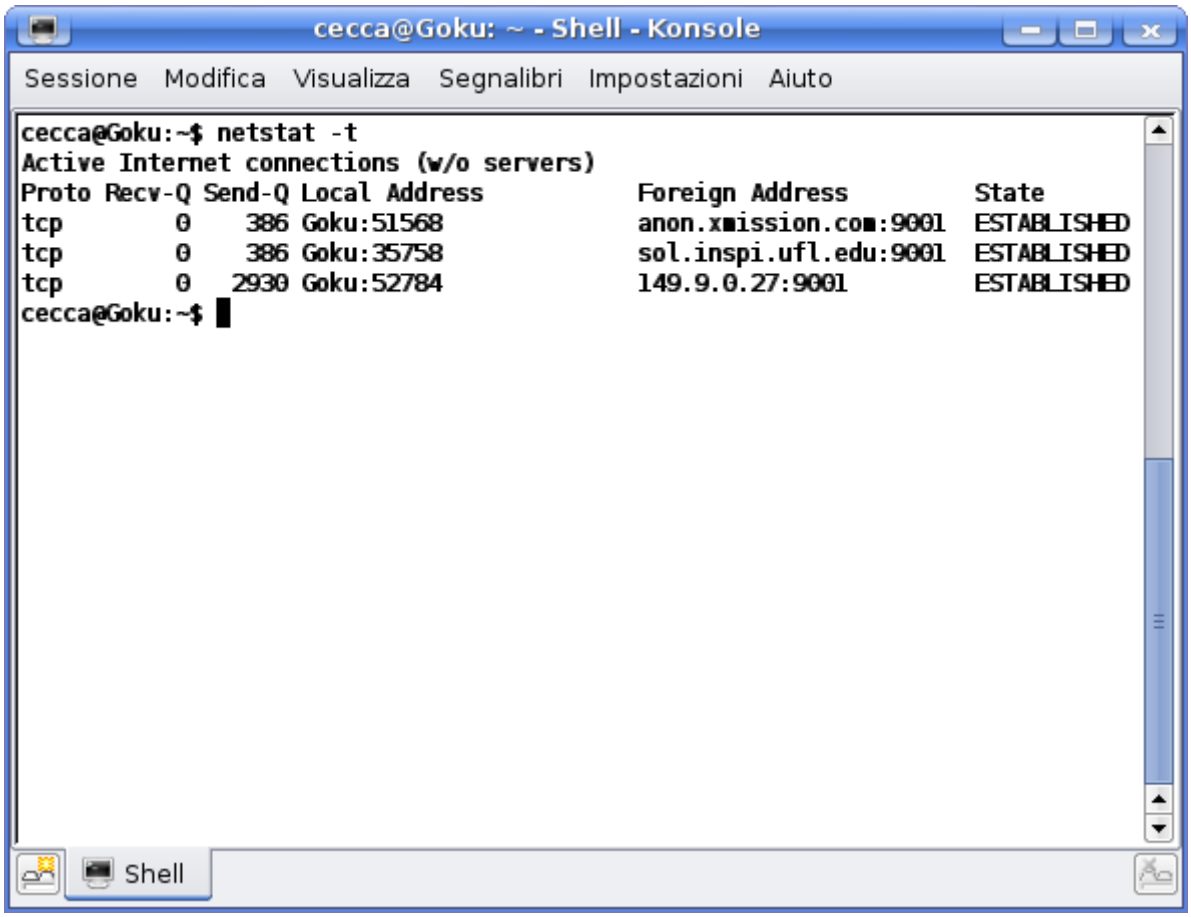
+	Bit 0–3	4–9	10–15	16–31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgment Number			
96	Header Length	Reserved	Flags	Advertise Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160 o 192+	Data			

- *Source Port*, che identifica il numero di porta sull'host mittente;
- *Destination Port*, che contiene il numero di porta sull'host destinatario;
- *Sequence Number*, ovvero il numero di sequenza, espresso in byte, dell'inizio del segmento TCP all'interno del flusso completo;
- *Acknowledgment Number*, cioè un numero di conferma che indica il Sequence Number del prossimo pacchetto TCP che l'host si aspetta di ricevere;
- *Header Length*, indica la lunghezza dell'intestazione del segmento TCP;
- *Reserved*, bit non ancora utilizzati che sono stati predisposti per sviluppi futuri del protocollo;
- *Flags*, che contiene i bit utilizzati per il controllo delle priorità e della gestione delle connessioni;
- *Advertise Window*, che indica la dimensione la finestra di ricezione dell'host mittente;
- *Checksum*, ovvero quel campo che viene utilizzato per la rilevazione dell'errore mediante la tecnica del checksum;
- *Urgent Pointer*, ovvero un puntatore che indica lo scostamento dal Sequence Number dei dati urgenti, cioè quei dati che hanno una priorità più alta degli altri;
- *Options*, che contiene informazioni usate per scopi avanzati del protocollo; questo campo è facoltativo.

Monitoraggio delle connessioni

Nei sistemi GNU/Linux le connessioni a livello trasporto possono essere monitorate attraverso l'uso del programma Netstat, che ci stampa a video una tabella con la situazione delle porte TCP, UDP e di altri protocolli, su cui però non spazieremo.
Le principali opzioni del comando sono:

- *-a*: che permette di vedere anche lo stato delle porte su cui non sono attive connessioni, ma su cui esiste un processo server in ascolto; di default infatti il comando stampa solo le porte su cui è attiva una connessione o su cui sta avvenendo un trasferimento dati;
- *-t*: che permette di vedere lo stato delle sole porte TCP;
- *-u*: che permette di vedere lo stato delle sole porte UDP;
- *-n*: che mostra gli indirizzi numerici, evitando di risolverli in nomi logici.



LIVELLO APPLICAZIONE

Nello stack TCP/IP il livello applicazione assume le funzioni che nel modello OSI sono svolte dai tre livelli superiori: sessione, presentazione ed applicazione. Questo livello perciò non ha solo il compito di fornire all'utente le applicazioni di rete da usare, ma fa suoi anche gli obiettivi degli altri due livelli, tra cui la sincronizzazione della comunicazione, la trascodifica, la compressione e la sicurezza dei dati. A seguire descriveremo i più usati protocolli di questo livello, assieme alle applicazioni che forniscono i rispettivi servizi in GNU/Linux.

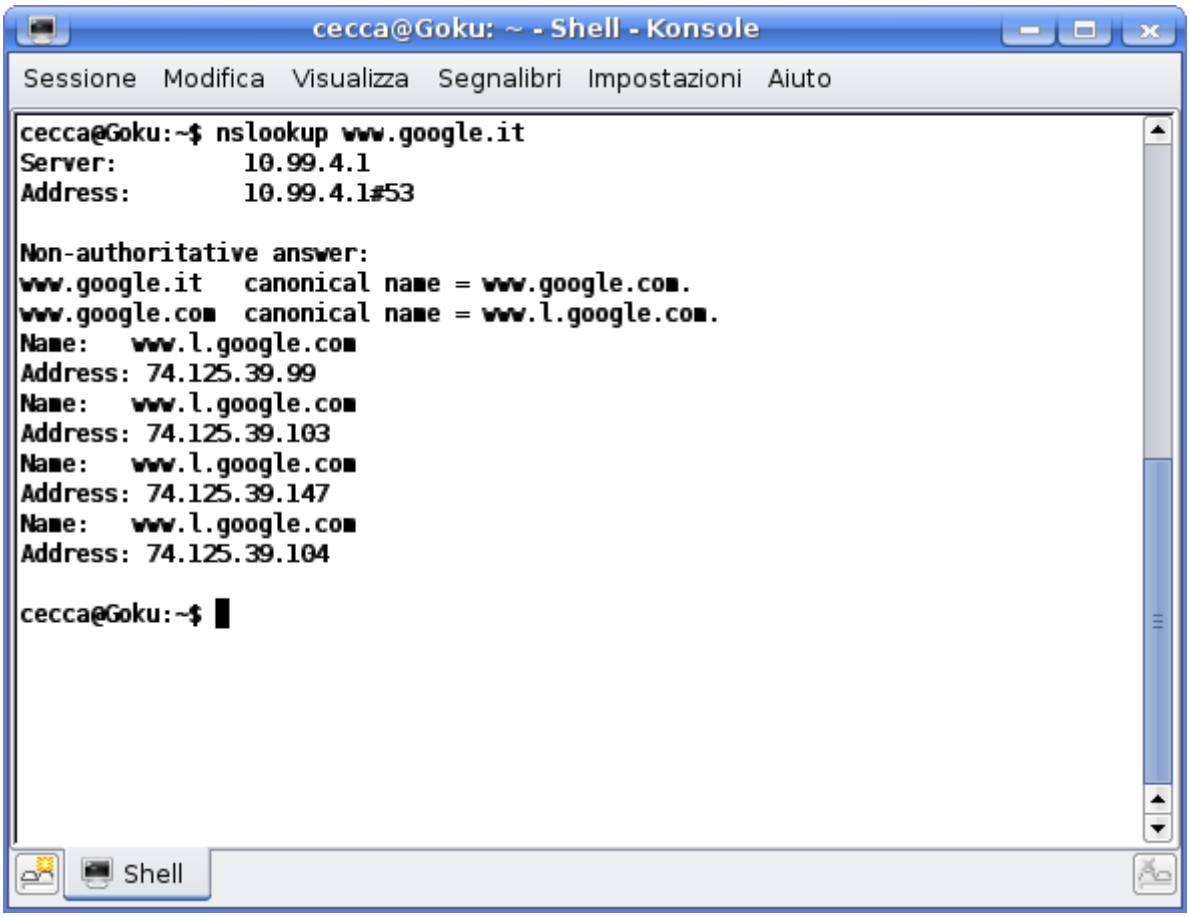
DNS

Come abbiamo precedentemente illustrato l'identificazione degli host è eseguita a livello rete con le funzioni di indirizzamento. A volte però è molto difficile ricordare mnemonicamente i vari indirizzi IP delle macchine server. A questo scopo è stato quindi creato il servizio di DNS (Domain Name System), che permette di associare ad ogni indirizzo IP un nome logico facilmente ricordabile. Per esempio quando contattiamo *www.google.it* non facciamo altro che contattare il server 66.249.93.104. Ma come facciamo a convertire, o meglio, risolvere il nome in un indirizzo? Come avevamo visto anche nella configurazione del server DHCP la rete ha bisogno di essere configurata con dei parametri, tra cui anche i server DNS. Questi sono in ascolto su host remoti alla porta 53 UDP e vengono contattati quando i nostri applicativi si trovano di fronte a un nome logico da risolvere in indirizzo. Se vogliamo però contattare e

risolvere un indirizzo, senza collegarci al server, possiamo utilizzare il comando *Nslookup* nella sintassi:

nslookup <nome logico server>

verrà così stampato a video la corrispondenza tra indirizzo IP e nome logico.



Altrimenti se vogliamo creare un piccolo sistema di risoluzione dei nomi locale possiamo procedere editando il file di testo */etc/hosts*. Qui ogni riga contiene una corrispondenza, usando la sintassi:

<indirizzo IP 1> <nome logico 1>
<indirizzo IP 2> <nome logico 2>
...
<indirizzo IP N> <nome logico N>

Per esempio aggiungendo al file la riga:

192.168.0.1 router

possiamo dare il comando:

ping router

e il nostro computer inizierà ad inviare pacchetti ICMP all'host 192.168.0.1.
I server DNS che il nostro sistema e' configurato per contattare sono elencati nel file */etc/resolv.conf* e possono essere dunque modificati o sostituiti in qualsiasi momento dall'utente root.
Infine è importante da ricordare che per chi non ha un indirizzo IP pubblico e statico esistono server che mettono a disposizione servizi di DDNS (Dynamic DNS), che permettono agli utenti di aggiornare la corrispondenza nel database del server DNS ogni volta che negoziano un IP diverso.

HTTP

Il protocollo HTTP (HyperText Tranfer Protocol) nasce nel 1991 al CERN di Ginevra, per opera di Tim Berners-Lee, al fine di sviluppare un metodo di condivisione delle informazioni tra le comunità mondiali dei fisici. Esso però ha trovato presto grande diffusione fra tutti gli utenti della rete Internet, così da divenire il protocollo che viene utilizzato da quello che è forse il servizio più diffuso sulla rete: il WWW (World Wide Web), ovvero un insieme di computer, detti server web, che mettono a disposizione dei client (browser) pagine speciali, scritte in un apposito linguaggio di scripting, detto HTML (HyperText Markup Language) e che possono contenere sia testi che immagini, suoni o video. Il trasferimento delle pagine dai server ai client è regolato proprio dal protocollo HTTP, che permette agli utenti di richiedere una pagina web ad un server specificando un URL (Uniform Resource Locator) del tipo:

http://<nome sito + percorso della risorsa nel computer remoto>

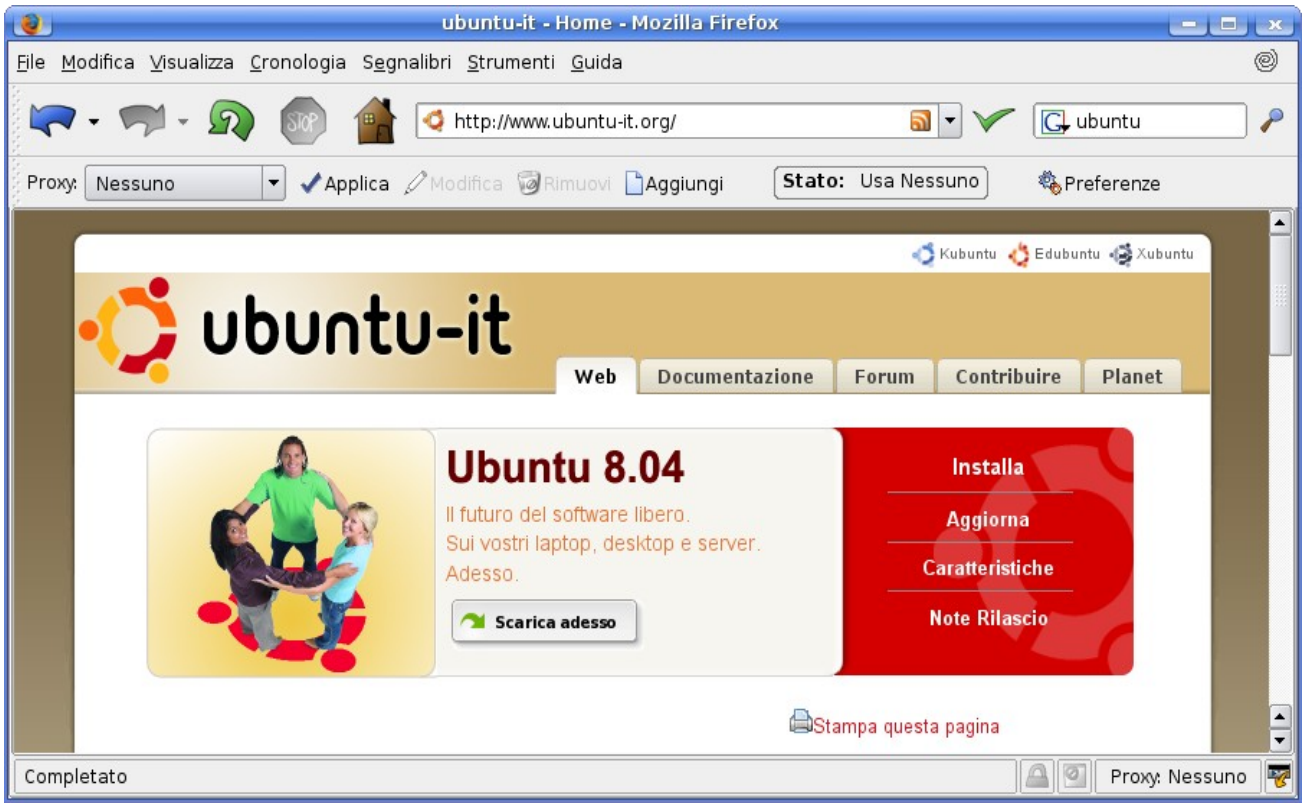
Per esempio qui vediamo in rosso il nome del server web e in blu il percorso:

http://cecca89.webhop.net/form/radio.htm

L'HTTP è un protocollo di tipo:

- *request/response* (richiesta/risposta), ciò vuol dire che ad ogni richiesta del client il server web risponde con un messaggio di risposta; esistono infatti due soli tipi di messaggio HTTP:
  - *messaggio di richiesta* composto da:
    - *riga di richiesta*, composta a sua volta da:
      - *metodo*, ovvero cosa il client chiede di fare (es. GET per ottenere una pagina e PUT per inviarne una);
      - *oggetto della richiesta*, che corrisponde al percorso dove trovare la pagina che vogliamo;
      - *versione del protocollo*, tipicamente HTTP/1.1;
    - *header*, che contiene informazioni aggiuntive sul browser utilizzato;
    - *body*, con il corpo del messaggio;
  - *messaggio di risposta*, composto a sua volta da:
    - *riga di stato*, che contiene un codice di tre cifre univoco per ogni tipo di risposta (es. 404 per una risorsa non trovata e 200 per un oggetto correttamente inviato al client);
    - *header*, che contiene informazioni aggiuntive sul server web e sull'oggetto trasferito;
    - *body*, con il corpo del messaggio;
- *one-shot-connection*, per cui ad ogni elemento che compone la pagina web, il browser apre una connessione sulla porta 80 TCP con il server e poi, alla fine del trasferimento, la chiude;
- *stateless*, poiché non sa mantenere lo stato dei dati che ha scambiato e non memorizza le informazioni transitate precedentemente tra browser e server web; questa carenza è però sopperita dal meccanismo delle sessioni, implementabile attraverso i cookie.

Il protocollo HTTP esiste anche in una versione criptata (detta HTTPS) usata soprattutto dai siti web che trasferiscono con il browser informazioni delicate, quali dati sensibili o numeri di carte di credito.
In GNU/Linux il browser più utilizzato è certamente Mozilla Firefox, anche se molto diffusi sono anche Opera, Konqueror o Epiphany.



Esiste inoltre anche un famoso browser testuale, per chi non utilizza nessuna interfaccia grafica, ma solo ed esclusivamente la shell. Esso si chiama Lynx, ma chiaramente non supporta nessun tipo di oggetto multimediale, quali immagini o video.
Invece, la soluzione server più utilizzata è certamente Apache. Per installare Apache su una distribuzione di derivazione Debian possiamo usufruire del package manager APT con il comando:

apt-get install apache2

Questo comando procede con il scaricare, installare e far partire come demone il server web. Successivamente per configurare il tutto è necessario editare il file di testo */etc/apache2/apache2.conf*, che appena installato contiene già una configurazione di default. Percorriamo dunque il significato delle più importanti direttive contenute nel file:

- *ServerRoot* specifica quale cartella utilizzare come directory principale per salvare file di configurazione, log e messaggi di errore;
- *Listen* indica su quale socket il processo server deve rimanere in ascolto; se specificata solo la porta Apache può essere contattato da qualsiasi interfaccia di rete;
- *DocumentRoot* specifica dove si trova la directory di root per le pagine web da servire ai browser;
- *ErrorDocument* consente di specificare un file di log dove memorizzare tutti gli errori in cui è incappato il server;
- *DirectoryIndex* indica quale è la pagina predefinita proposta dal server quando il browser richiede l'indice di una directory e non una pagina ben definita; per esempio quando digitiamo nella barra dell'indirizzo del browser *www.google.it*, solitamente il server ci restituisce la pagina *index.html*, che sta nella directory specificata in *DocumentRoot*.

Apache supporta inoltre anche il protocollo sicuro HTTPS.
Infine è importante puntualizzare che una configurazione del genere non ci consente di fornire ad un ipotetico browser pagine web dinamiche, il cui codice

HTML viene generato ad ogni richiesta in base ad alcuni script. Questi ultimi sono scritti in un linguaggio che, spesso, quando si parla di Apache è il PHP. Per integrare l'interprete PHP in Apache bisogna installare un modulo, che si occupa della traduzione del codice da linguaggio dinamico ad HTML. Utilizziamo dunque il seguente comando:

apt-get install libapache2-mod-php5

Dopo ciò saremo in grado di interpretare lato server pagine dinamiche e se volessimo agire sulla configurazione dell'interprete dovremmo editare il file di testo */etc/php5/apache2/php.ini*, dove possiamo trovare opzioni relative alla visualizzazione degli errori di interpretazione, alla gestione dei dati, o alle opzioni sulle estensioni dinamiche, cioè moduli che possiamo affiancare all'interprete PHP per estendere le sue funzionalità. Spesso infatti è molto utile avere un DBMS interfacciato con il server. A questo scopo installiamo anche MySQL, sia in versione server che client:

apt-get install mysql-client mysql-server

e infine installiamo le estensioni che permettono all'interprete PHP di interfacciarsi con il server MySQL:

apt-get install php5-mysql

A questo punto il server è pronto a fornire pagine web dinamiche, che possono anche contenere dati estratti da una base di dati.

Posta elettronica

Un altro tra i servizi più diffusi tra gli utenti di Internet è la posta elettronica (email). La posta elettronica permette di inviare messaggi ad altri utenti, che li riceveranno attraverso la loro casella di posta, ovvero uno spazio riservato all'utente su un server di ricezione dei messaggi.
La posta elettronica presenta i seguenti vantaggi rispetto ai normali servizi postali:

- la possibilità di allegare alla posta uno o più file, la cui dimensione può variare dal tipo di servizio messo a disposizione dal server;
- i tempi di arrivo del messaggio, che, nella maggior parte dei casi, sono praticamente immediati;
- la possibilità di inviare lo stesso messaggio a più di un destinatario.

La posta elettronica utilizza due protocolli per funzionare:

- *SMTP* (Simple Mail Transfer Protocol), che permette l'invio del messaggio attraverso una connessione TCP sulla porta 25 del server di invio; solitamente i client SMTP vengono configurati con il server SMTP che mette a disposizione il provider con cui è stato anche stipulato il contratto per l'accesso ad Internet;
- *POP3* (Post Office Protocol), che permette il download della posta sul computer locale, attraverso una connessione TCP con la porta 110 del computer in cui risiede la casella di posta; prima di inviare la lista dei messaggi ricevuti però il server POP3 richiede al client le credenziali dell'utente (username e password) per identificare la sua vera identità. Questo tipo di server assegna, ad ogni utente che ne fa richiesta, non solo una casella di posta, ma anche un indirizzo, del tipo:

  <utente>@<nome logico server>

  che dovrà essere specificato nelle mail a lui dirette.

Inoltre il POP3 sta per essere soppiantato da un ulteriore protocollo, che permette al client di consultare la posta senza scaricarla sul computer locale; questo protocollo è l'IMAP (Interactive Mail Access Protocol), che così facendo assicura all'utente la possibilità d visionare la propria posta anche da computer diversi.
In GNU/Linux vengono solitamente installati client di posta grafici che hanno la possibilità di comunicare attraverso tutti e tre i protocolli di posta visti. Il più diffuso è sicuramente Mozilla Thunderbird, che permette anche di mantenere sul PC una rubrica degli indirizzi e di filtrare i messaggi di posta indesiderati.

Esso è installabile con il comando:

apt-get install mozilla-thunderbird

e, per quanto riguarda la configurazione, necessita di:

- *indirizzo email*;
- *socket di un server POP3 o IMAP* in ascolto, in grado di far avere al client i messaggi che arrivano nella casella di posta remota;
- *credenziali* per accedere alla casella di posta (username e password);
- *socket di un server SMTP* in ascolto, in grado di inviare alla casella di posta del destinatario i messaggi generati dal programma;

Esistono poi altre opzioni che non sono però strettamente necessarie, ma ampliano le funzioni del programma permettendo all'utente di gestire i messaggi scaricati in cartelle o di apporre una firma digitale sui messaggi in uscita.
Da ricordare che solitamente i servizi di posta sono offerti proprio dai provider che forniscono connettività. Essi richiedono una registrazione via web e forniscono poi tutti i parametri per usufruire del servizio di posta dal client del computer di casa.
Infine esistono anche soluzioni per configurare il nostro PC GNU/Linux come un server POP o SMTP: tra le più diffuse ci sono rispettivamente Qpopper e Sendmail.

FTP

FTP (File Transfer Protocol) è un protocollo usato per trasferire file da un server ad un client o viceversa. Il servizio FTP può essere usato in due modi:

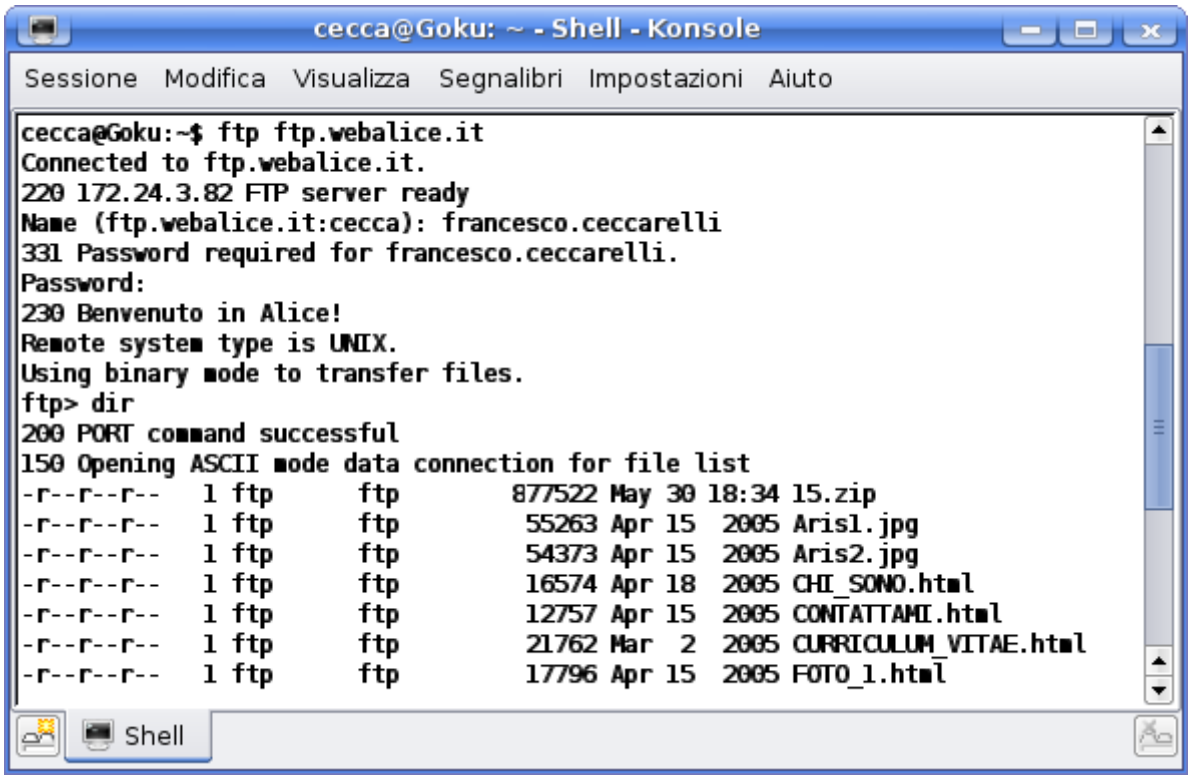
- *anonimo*, il più diffuso sula rete, che permette ad un client di connettersi al server senza identificarsi; in genere il client ha solo il permesso di scaricare file;
- *con login*, utilizzato dai server web che fanno hosting per permettere agli utenti di caricare le proprie pagine web, obbliga l'utente a inserire username e password per connettersi.

Ogni utente appena si collega ha l'opportunità di visionare il file system del server: l'utente però non ha una visione completa di tutti gli hard disk, ma viene relegato all'interno di una sola directory, che prende il nome di home directory e che rappresenta la sua root. Qui può avere, o non avere, gli stessi permessi tipici di Unix: lettura, scrittura, esecuzione.
Il protocollo FTP sfrutta una connessione TCP alla porta 20, per lo scambio dati, e alla 21, per i messaggi di controllo. Come d'altronde anche l'HTTP, questo protocollo non è criptato, perciò ne esiste anche una versione sicura detta SFTP. Infine un ulteriore versione del protocollo è il TFTP (Trivial FTP) che permette un trasferimento più veloce, ma meno affidabile, poiché esso viene incapsulato in UDP.
Ogni distro solitamente ha integrato il comando di shell *Ftp*, che permette il collegamento ad un server FTP remoto. La sintassi è:

ftp <indirizzo server>

che permette di arrivare alla fase di login. In caso il login fosse anonimo dobbiamo inserire come username *anonymous* e lasciare vuota la password; arrivati dunque alla console FTP qui dobbiamo digitare i comandi tipici del protocollo. I più importanti sono:

- *bye*, per disconnettersi;
- *cd <cartella>*, per cambiare cartella e spostarsi all'interno del filesystem remoto;
- *delete <file>*, per cancellare un file;
- *dir*, per elencare tutti gli oggetti presenti nella cartella in cui ci troviamo;
- *get <file>*, per scaricare un file (download);
- *help*, per visualizzare la lista di tutti i comandi disponibili;
- *put <file>*, per inviare un file nella cartella attuale (upload);
- *pwd*, per visualizzare il percorso dove ci troviamo;
- *rename <file>*, per rinominare un file;
- *rmdir <cartella>*, per cancellare una cartella.



Questo client ha però il grande svantaggio di essere un programma con interfaccia a linea di comando. Esistono dunque anche client FTP che utilizzano l'interfaccia grafica del sistema operativo, quali ad esempio Filezilla Client.
Per quanto riguarda invece il server FTP una delle implementazioni più usate è Vsftpd. Esso si installa con il comando:

apt-get install vsftpd

e si inserisce automaticamente tra i demoni del sistema. Per quanto riguarda la configurazione, essa è contenuta nel file */etc/vsftpd.conf*, le cui direttive più importanti sono:

- *anonymous_enable*, che, se impostato su *yes,* permette la connessione anonima al server;
- *local_enable*, che, se settato a *yes,* consente l'accesso per gli utenti del sistema operativo; questi una volta eseguito il login avranno gli stessi privilegi che avrebbero eseguendo un normale login sulla macchina server;
- *write_enable*, che serve ad abilitare per gli utenti l'upload oltre che al download, abilitato già di default;
- *anon_upload_enable*, che permette, se impostato su *yes,* agli utenti anonimi di caricare file sul server;
- *anon_mkdir_write_enable*, che, se settato a *yes,* consente agli utenti anonimi di creare directory sul server remoto.

Telnet & SSH

Telnet è un altro protocollo di tipo client-server che appartiene alla categoria dei protocolli di livello applicazione. Esso è solitamente utilizzato per avviare sessioni di lavoro su host remoti, con interfaccia a linea di comando. Ciò permette ad un utente di prendere il controllo di un computer remoto e lavorare su di esso proprio come se gli stesse seduto davanti. Il server telnet è solitamente in ascolto sulla porta 23 TCP e, in GNU/Linux, questo permette ai

client di aprire shell remote.
Il client telnet è installato di default su praticamente tutti i computer Unix, e quindi anche su tutte le distribuzioni GNU/Linux. Per usufruirne basta dare da shell il seguente comando:

telnet <indirizzo remoto>

Dopo il comando ci verrà chiesto nome utente e password, poiché per connettersi al server è strettamente necessario un account sul sistema remoto. A login avvenuto potremo accedere alla shell remota ed eseguire tutte le operazioni che avremo potuto eseguire sul posto, limitatamente ai permessi di cui dispone il nostro account; per ragioni di sicurezza spesso l'accesso remoto da root non è permesso.
Per installare un server remoto sulla nostra macchina è invece necessario installare il server telnet:

apt-get install telnetd

ma al contrario degli altri server esso non si autoconfigura come demone. è necessario infatti farlo partire automaticamente da root con il comando:

in.telnetd -debug

e questo si pone automaticamente in ascolto sulla porta 23 TCP.
Telnet fu però sviluppato quando ancora Internet non era altro che un insieme di università connesse fra loro, e il problema sicurezza non era ancora rilevante. Al giorno d'oggi chiunque può accedere ad Internet e così chiunque che si permette di usare una sessione telnet tra due host sulla rete Internet deve tenere ben presente che le informazioni scambiate (username e password compresi) transitano in chiaro e quindi qualsiasi persona potrebbe teoricamente intercettarle usando un analizzatore di rete. Capiremo meglio questo rischio nel prossimo capitolo relativo alla sicurezza. Per ora ci basta sapere che a causa di ciò il protocollo telnet è stato ampiamente soppiantato da SSH (Secure SHell), che si differenzia dal primo protocollo per il fatto di permettere connessioni non in chiaro, ma cifrate, sulla porta 22 TCP.
La più famosa implementazione di questo protocollo la possiamo trovare in OpenSSH, disponibile sia in versione server che in versione client, ma non sempre incluso in tutte le distribuzioni. Quindi per installare i software dobbiamo dare il comando:

apt-get install openssh-client

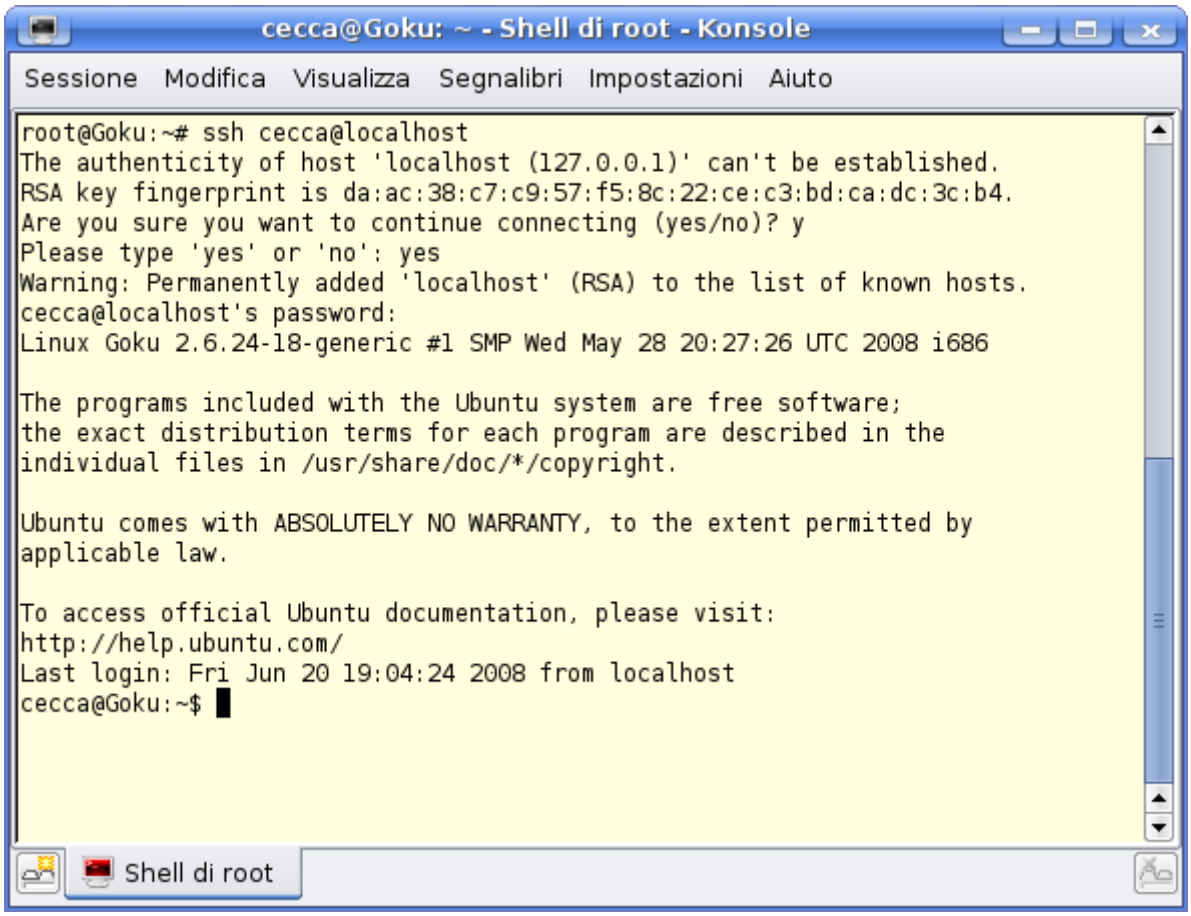
oppure:

apt-get install openssh-server

Il client può essere utilizzato subito appena completato download e installazione. La sintassi è:

ssh <utente>@<indirizzo remoto>

a questo punto sicuramente ci verrà chiesta la password che l'utente specificato ha impostato sul server remoto e poi possiamo cominciare ad utilizzare la nostra shell remota.



Il server invece, a differenza del server telnet, si installa tra i demoni del sistema e si pone fin da subito in attesa di connessioni sulla porta 22 TCP. Nel caso volessimo modificare le impostazioni predefinite del programma dobbiamo agire sul file */etc/ssh/sshd_config*, che ci permette di modificare opzioni quali:

- *la porta su cui il server è in ascolto*; infatti sapere che la porta 22 TCP è assegnata solitamente al server SSH potrebbe essere un vantaggio per una persona che voglia cercare di rompere il nostro server, così esso viene spostato su un'altra porta a piacere;
- *il tipo di autenticazione*, che può essere a password semplice, a chiave pubblica (RSA) o con l'utilizzo di un server Kerberos;
- *permettere oppure no il login come root*; conviene infatti disabilitare questa opzione, per evitare che qualcuno riesca a prendere il controllo del computer con privilegi amministrativi.

PARTE IV
La sicurezza

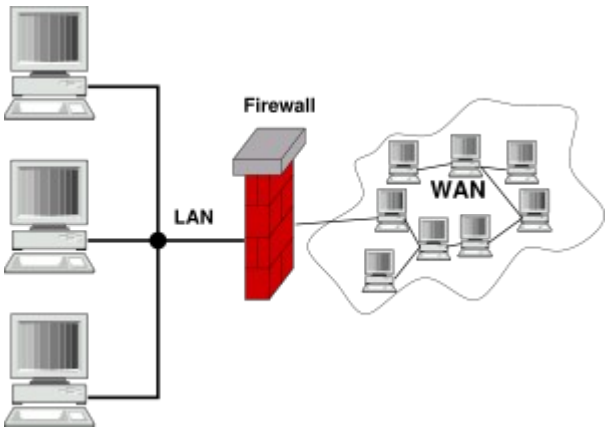
Un importante aspetto che gestisce il livello applicazione è quello della sicurezza dei dati. Internet è infatti una rete ad estensione mondiale e ad accesso pubblico: il che vuol dire che chiunque può connettersi ad essa e tentare di mettersi in comunicazione con il nostro computer, anche per scopi non sempre benevoli.
La sicurezza dei nostri dati è definita secondo quattro parametri:

- *disponibilità*, poiché deve sempre essere garantito l'accesso ai dati da parte di chi ne ha il permesso;
- *riservatezza*, poiché essi non devono poter essere accedibili da parte di chi non ha le giuste credenziali;
- *originalità*, poiché essi devono sempre presentarsi all'utente senza modifiche non autorizzate;
- *integrità*, poiché devono essere protetti da ogni minaccia, anche naturale, che cerchi di comprometterne o corromperne il contenuto.

In particolare in questo capitolo vedremo alcuni esempi su come difenderci dalla minaccia rappresentata dall'uomo, che spesso si concretizza sotto la forma di un attacco informatico, ovvero un tentativo di accesso non autorizzato al sistema, che possa mettere a rischio uno dei parametri sopra descritti.

FIREWALL

Un firewall è un dispositivo, hardware o software, che si colloca fra due o più reti, permettendo di controllare e filtrare il traffico che lo attraversa. Solitamente esso viene utilizzato tra due reti che hanno livelli di sicurezza diversi, in particolare tra una rete locale sicura e la rete Internet. Per far ciò i firewall, come i router, dispongono di più interfacce, ad ognuna delle quali può essere collegata una rete diversa.



I firewall vengono classificati come:

- *a filtro di pacchetto* (packet filter): essi possono lavorare solo a livello rete e trasporto, quindi non sono in grado di filtrare il traffico in base a ciò che è realmente contenuto dai pacchetti; essi possono infatti eseguire filtri in base a ciò che riportano le intestazioni dei protocolli di questi livelli, quindi in base a indirizzi IP sorgente e destinazione, porta, anch'essa sorgente e destinazione, e tipo di protocollo; i filtri possono essere di tre specie:
  - *input*, che si applicano ai pacchetti diretti al firewall stesso;
  - *output*, che si applicano invece a pacchetti in uscita dal firewall;
  - *forward*, che si applicano a quei pacchetti che devono attraversare il firewall e che esso deve instradare verso la giusta interfaccia.

E' possibile impostare i filtri creando delle regole; queste regole possono essere regole di accettazione oppure possono essere di rifiuto, mentre per i pacchetti che non corrispondono a nessuna regola sono stabilite politiche di default; è possibile inoltre eseguire un ulteriore classificazione dei firewall a filtro di pacchetto:

- *stateless*, che non riescono a tenere traccia delle connessioni TCP e non riescono dunque ad individuare pacchetti malevoli che non appartengono a nessuna connessione aperta volontariamente dall'host; molti router hanno un firewall di questo tipo integrato;
- *stateful*, che sono invece in grado di tenere traccia delle relazioni tra i pacchetti che lo attraversano e possono dunque ricostruire lo stato delle connessioni attive in un determinato momento; essi sono in grado di individuare quei segmenti TCP che vengono inviati all'host allo scopo di ottenere informazioni dal sistema.

- *Application layer firewall*, che sono in grado di interpretare il contenuto dei segmenti TCP, permettendo di riconoscere virus e di impostare regole sui dati trasportati; un esempio di essi lo troviamo nei server proxy, che nella maggior parte delle volte sono specifici per un protocollo e possono filtrare i dati che li attraversano in base a parole chiave o regole specifiche del protocollo; spesso infatti gli HTTP proxy sono configurati per frapporsi tra una rete privata e Internet, in modo tale da filtrare contenuti pornografici o comunque non adatti all'utenza della rete privata. Questo tipo di firewall però ha lo svantaggio di aumentare la complessità dei controlli, rallentando il traffico.

In particolare tratteremo d'ora in poi i firewall a filtro di pacchetto, che principalmente vengono implementati nei seguenti modi:

- *integrati nei router*: esistono infatti router che mettono a disposizione una ACL (Access Control List) che permette all'amministratore di creare le regole sopra descritte in funzione di liste di indirizzi IP o di indirizzi MAC;
- *firewall hardware*: esistono infatti soluzioni hardware costruite appositamente per controllare il traffico tra due reti. Esso va montato tra router e LAN privata;
- *computer con firewall*: è un normale computer con tante schede di rete quante sono le interfacce di cui ha bisogno; esso dispone di un firewall software installato, oppure di un sistema operativo costruito appositamente per svolgere questa funzione (es. IPCop);
- *personal firewall*: è un firewall software dedicato a soluzioni domestiche; esso si installa su un PC ed ha il compito di proteggere il singolo PC da tutto ciò che sta fuori; questo è l'unico firewall che è in grado di creare regole anche in base a processi in esecuzione sulla macchina.

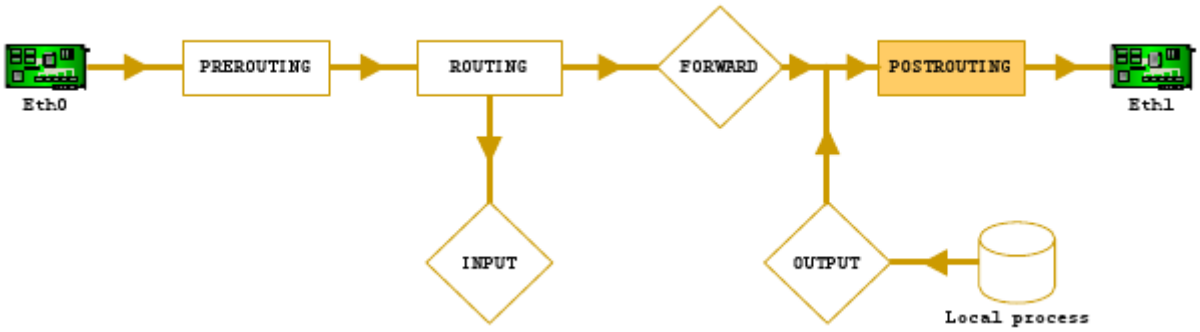
In Linux è stato implementato, proprio all'interno del kernel, un sistema che permette il filtraggio stateful dei pacchetti, con cui è però anche possibile realizzare funzioni di NAT, al fine di modificare gli indirizzi dei pacchetti in transito nel sistema. Il nome dato a questa componente è Netfilter, ma la configurazione dei filtri viene eseguita, con permessi di root, attraverso il comando *Iptables*.

In Netfliter i filtri sono detti regole; le regole sono raggruppate in catene e le catene sono a loro volta raggruppate in tabelle. Ogni tabella si occupa di raggruppare catene riguardanti uno stesso servizio, mentre le catene sono ACL, ognuna delle quali rappresenta una diversa fase dell'elaborazione del pacchetto; le regole contenute nella catena vengono dunque applicate quando la fase associata ad essa viene attraversata. Inoltre ad ogni catena è associata una politica predefinita, che sancisce se i pacchetti che attraversano la catena

stessa devono essere bloccati o lasciati passare.
Ogni regola è composta da due parti principali: le caratteristiche che un pacchetto deve avere per applicarvi la regola (match) e l'azione da compiere (target o obiettivo). L'amministratore ha l'opportunità di aggiungere o rimuovere regole, catene o tabelle, ma appena installato il sistema appare diviso in tre tabelle, senza ancora regole:

- *tabella filtro*, da cui passano tutti i pacchetti e che permette di filtrarli in base alle regole impostate nelle catene; le catene sono tre:
  - *INPUT*, da cui passano tutti i pacchetti che hanno come destinazione l'host su cui è montato il firewall;
  - *OUTPUT*, da cui passano tutti i pacchetti che partono dall'host;
  - *FORWARD*, da cui passano tutti i pacchetti in transito sull'host e che questo instrada verso un'altra interfaccia;
- *tabella NAT* (Network Address Translation), che si occupa invece del servizio di traduzione degli indirizzi, sorgente o destinazione, che permette al firewall di modificare l'intestazione IP. Essa contiene tre catene:
  - *PREROUTING*, che viene applicata prima che l'host consulti la tabella di instradamento. Viene solitamente utilizzata per tradurre l'indirizzo destinazione (Destination NAT o DNAT);
  - *POSTROUTING*, che viene applicata invece subito dopo che l'host consulti la tabella di routing; di solito viene usata per tradurre l'indirizzo sorgente dei pacchetti (Source NAT o SNAT);
  - *OUTPUT*, che consente di fare NAT destinazione sui pacchetti in partenza dall'host stesso;
- *tabella mangle*, che permette di intervenire sulle intestazioni dei pacchetti, modificandone i flag. Le catene di questa tabella sono:
  - *PREROUTING*, che, come detto precedentemente, viene attraversata prima dell'instradamento, e in questo caso se il pacchetto è diretto al sistema viene passato alla catena INPUT, altrimenti viene instradato verso un'altra interfaccia (catena FORWARD);
  - *POSTROUTING*, che corrisponde all'omonima catena vista prima e a cui arrivano i pacchetti sia dalla catena FORWARD che da quella OUTPUT;
  - *INPUT*, ove passano i pacchetti in ingresso diretti all'host stesso;
  - *OUTPUT*, dove passano i pacchetti che l'host invia;
  - *FORWARD*, che rappresenta la fase intermedia tra PREROUTING e POSTROUTING, in cui i pacchetti vengono instradati;

Riassumendo i pacchetti vengono sottoposti alle regole contenute nelle catene con il seguente ordine:



La tabella che più ci interessa ai fini della sicurezza è quella *filtro*. Per modificare le regole contenute nelle sue catene usiamo la sintassi:

iptables { -A | -D } <catena> <regola> [opzioni]

dove l'opzione *-A* indica che vogliamo aggiungere una regola, mentre *-D* serve a cancellarne una esistente; da notare come non è necessario specificare la tabella, poiché la tabella filtro è quella predefinita. Inoltre il parametro regola va specificato solo nel caso volessimo cancellarne una, poiché aggiungendo una regola questa prende come identificativo un numero che si autoincrementa ogni volta.
Le principali opzioni sono divise in due parti, come le regole. Le seguenti permettono di stabilire il match:

- *-s <sorgente>*, che permette di specificare come sorgente un indirizzo IP o un indirizzo di rete in notazione CIDR;
- *-d <destinazione>*, che similmente all'opzione sopra descritta ci permette di specificare la destinazione;
- *--sport <sorgente>*, che indica la porta sorgente del pacchetto;
- *--dport <destinazione>*, che indica invece la porta destinazione del pacchetto;
- *-p <protocollo>*, che consente di stabilire il protocollo che deve essere sottoposto a filtraggio. Accetta come parametri *udp*, *tcp*, *icmp* o *all*;
- *-i <interfaccia>*, per specificare un'interfaccia di ingresso;
- *-o <interfaccia>*, per specificare un'interfaccia di uscita;

Per quanto riguarda invece il target, esso viene espresso con l'opzione:

-j <target>

dove i principali obiettivi sono:

- *ACCEPT*, che permette al pacchetto di arrivare, uscire o essere inoltrato dal sistema;
- *DROP*, che blocca il pacchetto senza avvisare il mittente;
- *REJECT*, che blocca il pacchetto, ma invia al mittente un messaggio di errore ICMP;

Per modificare invece la politica di default di una catena utilizziamo il comando:

iptables -P <catena> <target>

Infine per visualizzare tutte le regole contenute in una catena usiamo:

iptables -L <catena>

A volte risulta però scomoda, sopratutto per le configurazioni domestiche, l'interfaccia da shell. Così sono nate diverse implementazioni che forniscono front-end per *Iptables*, molto più intuitivi e user-friendly. Un esempio è il diffusissimo Firestarter.

PORT-SCANNING

La fase più importante di un attacco è sempre la raccolta di informazioni sulla vittima, al fine di scovare le debolezze del sistema da attaccare. Esistono molti metodi che ci permettono di raccogliere informazioni sull'obiettivo, tra cui possiamo annoverare metodi informatici e non, quali l'analisi dei rifiuti o la richiesta di informazioni all'utente sotto travestimento. Per quanto riguarda i metodi informatici spesso però questi sono gli stessi che può utilizzare l'amministratore di sistema per individuare le proprie vulnerabilità. A questo scopo uno dei tool più completi è Nmap. Esso ci permette di effettuare scansioni degli host appartenenti ad una rete (port-scanning), scoprendo le porte su cui è possibile effettuare una connessione ad un processo server.

Il risultato di una scansione con Nmap è una lista di porte, il loro stato e il servizio a cui essa è solitamente assegnata. Infatti per quanto riguarda lo stato una porta può essere:

- *aperta* (open), quando sulla porta è in ascolto un processo con cui è possibile aprire connessioni;
- *filtrata* (filtered), quando non possiamo fare nessuna affermazione sulla porta, poiché i dati sono filtrati da un firewall o da un altro sistema di sicurezza;
- *chiusa* (closed), quando sappiamo che su questa porta non ci sono processi in ascolto e quindi non è possibile aprirvi connessioni.

Anche se per Nmap esistono molti front-end che cercano di renderne l'utilizzo più semplice e intuitivo, noi tratteremo sempre e solo il suo utilizzo da shell. La sintassi base per eseguire scansioni è la seguente:

nmap [tipo di scan] [opzioni] {host o rete}

Dove *tipo di scan* sta per la modalità con cui viene effettuata la scansione e risponde ad una delle seguenti opzioni:

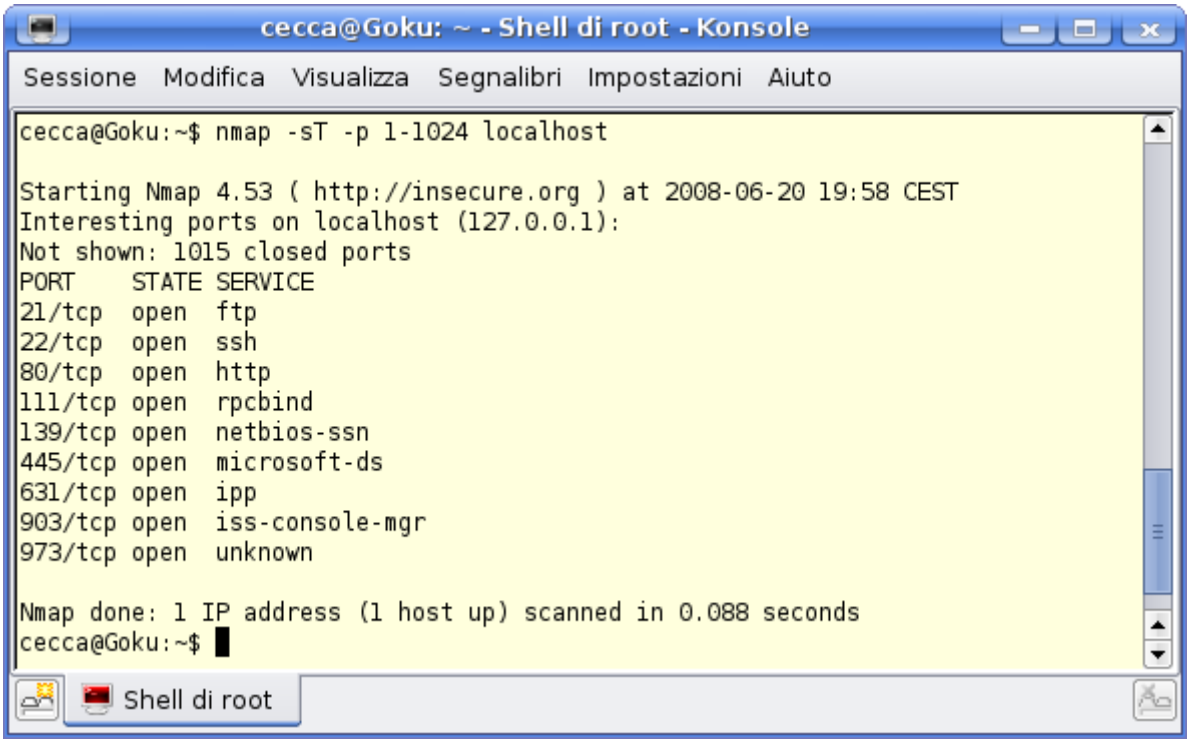
- *-sT* (TCP connect), rappresenta lo scan di base. Esso cercherà di stabilire una connessione TCP completa con ogni porta dell'host remoto specificato. Se la porta è in ascolto, la connessione avrà successo e la porta verrà segnalata come aperta, altrimenti come chiusa.
- *-sS* (TCP SYN), anche conosciuto come scanning half-open, perché non stabilisce una connessione TCP completa, ma manda una richiesta di connessione all'host remoto; se questo risponde con un acknowledgment vuol dire che la porta è aperta, se invece risponde con un pacchetto di rifiuto, la porta è chiusa. A questo punto la connessione sarà comunque terminata. Questo attacco è utilizzato quando vogliamo evitare che un'ipotetica vittima che mantiene un log delle connessioni ci individui; per questo attacco pero' sono necessari i privilegi di root;
- *-sA* (ACK scan), che consiste nel mandare un pacchetto ACK alle porte specificate. Se riceviamo un pacchetto di rifiuto la porta è classificata come non filtrata, mentre se non riceviamo niente, oppure riceviamo un errore ICMP, la porta è classificata come filtrata. Questo tipo di scan ha il difetto di non mostrare quali porte sono aperte;
- *-sP* (ping scanning), utile a capire quali host sono attivi nella rete che vogliamo verificare. Normalmente, questo scan avviene mandando un ICMP Echo Request ad ogni indirizzo IP della rete specificata;
- *-sU* (UDP scan), che serve per individuare quali porte UDP sono aperte su un host. Per fare ciò il programma manda dei pacchetti UDP senza payload all'host specificato. Se riceviamo un messaggio ICMP di errore, vuol dire che la porta è chiusa, altrimenti supponiamo che questa sia aperta.

Le principali opzioni specificabili sono:

- *-P0*, che permette anche la scansione di host che non rispondono alle richieste ICMP, poiché evita di mandare un ping prima di effettuare la scansione;
- *-PT*, che per determinare gli host attivi manda un TCP ACK al posto di un pacchetto ICMP;
- *-PS*, che usa un pacchetto di apertura della connessione TCP per determinare gli host attivi;
- *-PI*, che attraverso un ICMP Echo Request riesce a determinare gli host attivi;

- *-PB*, il tipo di ping predefinito, che usa un meccanismo combinato tra TCP ACK e richieste ICMP;
- *-O*, che permette di rilevare il sistema operativo della vittima, grazie alla tecnica del TCP\IP fingerprint;
- *-p <intervallo di porte>*, che permette di specificare l'intervallo di porte, o la porta, da scansionare;
- *-S <indirizzo IP>*, che viene usato per cambiare l'indirizzo IP sorgente dei pacchetti (IP spoofing);
- *-e <interfaccia>*, che precisa quale interfaccia utilizzare per effettuare le scansioni;
- *-g <numero di porta>*, che permette di scegliere la porta sorgente da cui vengono effettuate le connessioni TCP.

Infine per quanto riguarda l'ultimo campo possiamo passare al programma l'indirizzo IP di un solo host o addirittura anche di un insieme di host, utilizzando la notazione CIDR.



SNIFFING

Una volta individuate quali sono le porte di accesso al sistema, e quali servizi possono essere sfruttati per connettersi ad esso, la fase di raccolta delle informazioni non è ancora finita. Spesso infatti per sfruttare un servizio che abbiamo individuato con Nmap abbiamo bisogno delle giuste credenziali. Supponiamo dunque che se sul sistema della vittima sono attivi certi servizi, questi siano utilizzati da qualcuno. Se poi noi abbiamo in qualche modo accesso ad un computer collegato allo stesso hub della vittima (hub, non switch, a causa della sua natura broadcast) o abbiamo accesso ad un nodo intermedio della rete per il quale passano le informazioni dirette al computer da attaccare, possiamo pensare di intercettare le informazioni che questo scambia con gli altri computer. Per far ciò è necessario un analizzatore di rete, anche conosciuto come sniffer, che ci permetta di intercettare tutti i pacchetti che passano per la

nostra stazione. In GNU/Linux la più famosa implementazione è data da Tcpdump.
Per installare Tcpdump usiamo come al solito APT:

apt-get install tcpdump

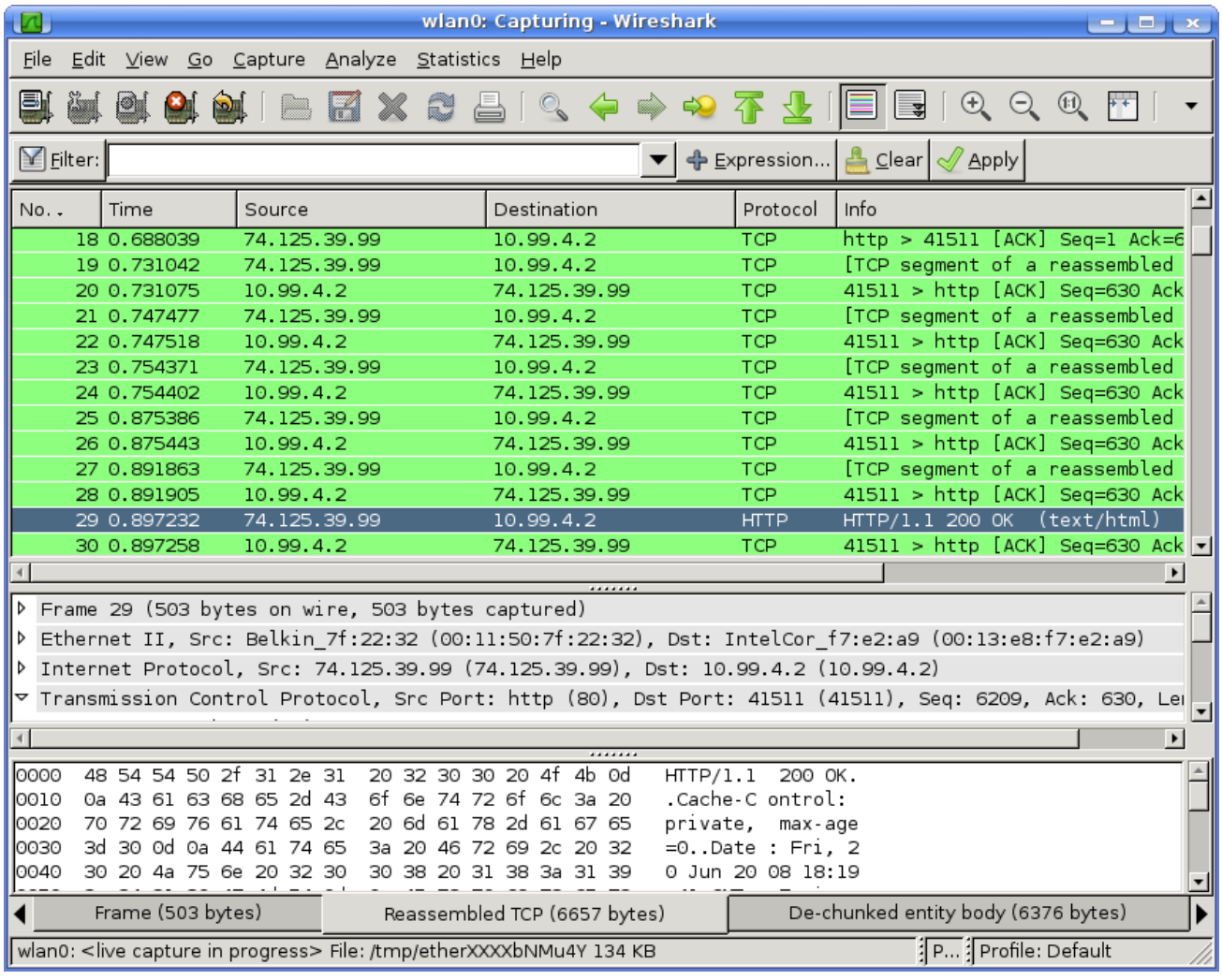
Per poi avviare il programma usiamo la seguente sintassi con privilegi amministrativi, che ci stamperà a video tutti i pacchetti intercettati:

tcpdump [-i <interfaccia>] [<filtro>]

dove l'interfaccia indica la scheda di rete installata sul computer da cui intercettare il traffico, mentre per filtro intendiamo un espressione booleana con cui stampare a video solo i pacchetti che rispettano certi requisiti. Le principali primitive che ci permettono di far ciò sono:

- *host <indirizzo IP>*, che assume valore vero solo se i pacchetti coinvolgono un certo host della rete;
- *ether host <MAC address>*, che si comporta come la primitiva precedente, ma in base al MAC address fornito;
- *port <porta>*, che restituisce un valore vero solo per i pacchetti che interessano una porta specificata;
- *ether proto <protocollo>*, che assume valore vero se il protocollo specificato è lo stesso incapsulato nel frame Ethernet intercettato;
- *ip proto <protocollo>*, che funziona come la precedente, ma fa riferimento al protocollo incapsulato in Ipv4;

Gli operatori sopra elencati sono poi collegati in un'unica espressione booleana dagli operatori logici *and*, *or*, oppure *not*, come in una qualsiasi espressione logica.



Infine esiste anche per Tcpdump un front-end: Wireshark, che ci permette di aprire i pacchetti leggendo tutte le intestazioni che li compongono, fino a leggere i messaggi a livello applicazione. Questa capacità risulta dunque molto pericolosa per i programmi che scambiano informazioni in chiaro, come ad esempio *Telnet*, che non si preoccupa di crittografare le credenziali che invia al server per le autenticazioni, o per i protocolli di posta elettronica, che inviano mail come un insieme di caratteri ASCII, non criptati.
Abbiamo dunque dimostrato come è semplice intercettare informazioni dalla rete e speriamo di aver fatto capire al lettore l'importantissimo ruolo che la crittografia dei dati gioca in tutto ciò.