Server software
Principali funzioni dei componenti software di un server di rete

Un *server* è composto principalmente da un sistema operativo, protocolli e moduli di rete e il supporto di programmi *server* e servizi.

Il sistema operativo

Le caratteristiche essenziali di un sistema operativo orientato alla gestione di ambienti distribuiti di rete sono la multiprogrammazione e la multiutenza.
Un sistema operativo multiprogrammato è in grado di eseguire più programmi contemporaneamente mediante algoritmi di gestione dei processi. Questa peculiarità è fondamentale per consentire ai *client* di utilizzare una moltitudine di programmi e servizi residenti sul *server*. La capacità di gestire più utenti consente una maggiore sicurezza del sistema unita possibilità di regolare permessi di accesso alle risorse.

Protocolli di rete

Il sistema operativo comprende inoltre moduli che implementano i protocolli di rete, possono essere integrati nel *kernel* oppure essere caricati o installati su richiesta. (Ad esempio moduli del *kernel* di *Linux* o l'installazione dei protocolli nella rete per i sistemi *Windows*).

Programmi *server* e servizi

Tutte le funzionalità *server* solitamente sono demandate a programmi specifici, denominati programmi *server*. Tali programmi vanno configurati e impostati secondo le esigenze della rete in cui operano, occorrono perciò buone conoscenze di base sia del sistema operativo in uso, sia delle reti e dei protocolli di rete utilizzati.
I programmi *server* installati implementano soltanto il lato *server* nell'architettura *client/server* necessaria per il funzionamento del servizio. Di solito, dipendentemente delle risorse del sistema (velocità di calcolo e spazio disco e banda disponibile), si integrano più programmi *server* sulla stessa macchina, dando luogo a sistemi concentrati di servizi di rete.

Confronto tra le funzioni dei principali sistemi operativi di rete

Esistono attualmente disponibili sul mercato o liberamente reperibili su Internet una grande quantità di sistemi operativi con implementazioni e funzionalità orientate alle reti.

Tra i sistemi maggiormente diffusi di tipo *open source* citiamo *GNU/Linux*: disponibile in una moltitudine di distribuzioni è implementazione libera e *open source* del sistema *UNIX*. La famiglia di sistemi BSD come *OpenBSD*, *NetBSD*, *FreeBSD* sono implementazioni *open source* derivate da BSD *UNIX*, la versione di *UNIX* sviluppata alla *Berkeley Univesity*.

Tra i sistemi operativi commerciali troviamo implementazioni del sistema *UNIX* come *Digital UNIX* (*Digital*), Tru64 (*Compaq*), HP-UX (HP), AIX (IBM) e Solaris (*Sun*). *Windows* NT/2000 e XP sono implementazioni del sistema *Windows* prodotto da *Microsoft* per le reti.

Tuttavia, per la generalità degli argomenti trattati e per la preponderante diffusione in ambienti scolastici presentiamo le configurazioni e le descrizioni dei sistemi operativi: *GNU/Linux RedHat* e *Microsoft Windows* 2000. Per le diverse versioni e varianti di questi sistemi operativi e per sistemi simili come struttura e funzionalità rimandiamo alle descrizioni ed ai manuali relativi.

Vedremo le loro principali strutture e funzionalità.

Caratteristiche di Windows2000 - Client

Microsoft Windows 2000 *Professional* migliora le caratteristiche delle precedenti versioni di *Windows* nelle seguenti aree aree:

- **Semplicità d'uso**. Oltre ai miglioramenti relativi all'interfaccia utente, la semplicità d'uso di *Microsoft Windows* 2000 *Professional* si traduce nelle funzionalità di Supporto per Utenti Remoti e nelle funzionalità di Supporto della Stampa. Per quel che riguarda il Supporto per Utenti Remoti ricordiamo le seguenti funzionalità:
  - *Network Connestion Wizard*. Permette di configurare tutte le tipologie relative ad un *client* di accesso remoto (connessione *dial-up*, connessione ad Internet, connessione VPN, connessione via cavo).
  - Supporto per *Virtual Private Network* (VPN). Permette di connettersi alla rete aziendale tramite un ISP in maniera sicura.
  - Cartelle *Offline*. Permette di memorizzare localmente *files* residenti su un *server* in maniera tale che siano accessibili anche in modalità *offline*.
  - Per quel che concerne invece il Supporto della Stampa ricordiamo le seguenti funzionalità:
    - *Internet Printing Protocol* (IPP). Permette di inviare documenti ad un *print server Microsoft Windows* 2000 tramite Internet.
    - *Add Printer Wizard*. Semplifica il processo di installazione di un *client*.
- **Gestione Semplificata**. Permette all'utente di gestire in maniera semplificata i dati, le applicazioni e tutte le impostazioni di sistema. Ricordiamo le seguenti funzionalità:
  - *ADD/Remove Programs Wizard*.
  - Servizio di *Windows Installer*.
- **Supporto *Hardware***. *Microsoft Windows* 2000 *Professional* supporta più di 7000 *device hardware*. Ricordiamo a tal proposito:
  - *Wizard Add/Remove Hardware*.
  - Supporto *Plug and Play*.
  - Opzioni di risparmio energetico: *standby*, ibernazione.
  - Sistema *multiprocessor* simmetrico (SMP). Fino a due processori.
- **Gestione dei *Files***. *Microsoft Windows* 2000 *Professional* supporta le seguenti tecnologie per semplificare e migliorare la gestione dei *files*:
  - *File System* NTFS. Supporta la crittografia, la sicurezza locale ed il controllo delle quote.
  - *File System* FAT32. Garantisce la piena compatibilità con *Microsoft Windows* 95 OSR2 e versioni successive.
  - *Utility* di Deframmentazione dei Dischi.
  - *Utility* di *Backup*.
- **Sicurezza**. *Microsoft Windows* 2000 *Professional* è il sistema operativo *desktop* più sicuro, sia per ambienti *stand-alone* che per ambienti di rete. Ricordiamo le seguenti funzionalità:
  - *Kerberos* 5. Protocollo standard di autenticazione che permette una autenticazione ed un accesso alle risorse di rete più efficaci.
  - *Encrypting File System* (EFS). Permette di cifrare i *file* memorizzati su disco.
  - *Internet Protocol Security* (IPSec). Permette di specificare politiche di cifratura per flussi di dati sulla rete.

Chiudiamo questa breve panoramica elencando quelli che sono i prerequisiti *hardware* per l'installazione di *Microsoft Windows* 2000 *Professional*:

- CPU: *Pentium based*
- Memoria : 32 MB (raccomandati 64 MB)
- Spazio Disco : uno o più dischi con un minimo di 650 MB (raccomandati 2 GB)
- Sulla partizione che conterrà i *files* di sistema

- Scheda di rete
- Scheda Video VGA Standard o superiore
- Tastiera e *Mouse*

Caratteristiche di Windows2000 - Server

Microsoft Windows 2000 *Server* è una piattaforma che contiene tutte le funzionalità di *Microsoft Windows* 2000 *Professional* più le funzionalità che ne ottimizzano le *performance* per le funzionalità di *file server*, *print server* ed *application server*.

Alla base ci sono tutta una serie di servizi basati su *Active Directory*, che permette di centralizzare la gestione di utenti, gruppi, sicurezza e risorse di rete.

Microsoft Windows 2000 *Advanced Server* supporta fino a 8 processori SMP e fino a 8 GB di memoria RAM.

Microsoft Windows 2000 *Server* è una buona soluzione per l'implementazione di soluzioni *enterprise* in realtà medio-piccole: *file server*, *print server*, *Web server*, *Terminal Services server*, *server* di accesso remoto.

Di seguito i requisiti *hardware* minimi per l'installazione di *Microsoft Windows* 2000 *Server*:

- Processore: 32-bit *Pentium* 133 MHz.
- Memoria: 64 MB per reti di meno di 5 *computer*; 128 MB è il minimo raccomandato per tutte le altre situazioni.
- Spazio Disco : uno o più dischi con un minimo di 680 MB (raccomandati 2 GB) Sulla partizione che conterrà i *file* di sistema.
- CD-ROM o DVD-ROM *drive* (El Torito-compatible).
- Scheda di rete.
- Scheda Video VGA Standard o superiore.
- Tastiera e *Mouse*.

Caratteristiche di Windows2000 - Windows 2000 Advanced Server and Datacenter Server

Microsoft Windows 2000 *Advanced Server* è la piattaforma *software* raccomandata per *application server* dipartimentali che necessitano anche di elevata disponibilità e bilanciamento dei carichi di lavoro (*cluster*). *Microsoft Windows* 2000 *Server* supporta fino a 8 processori SMP e fino a 8 GB di memoria RAM.

Di seguito i requisiti *hardware* minimi e raccomandati per l'installazione di *Microsoft Windows* 2000 *Advanced Server*:

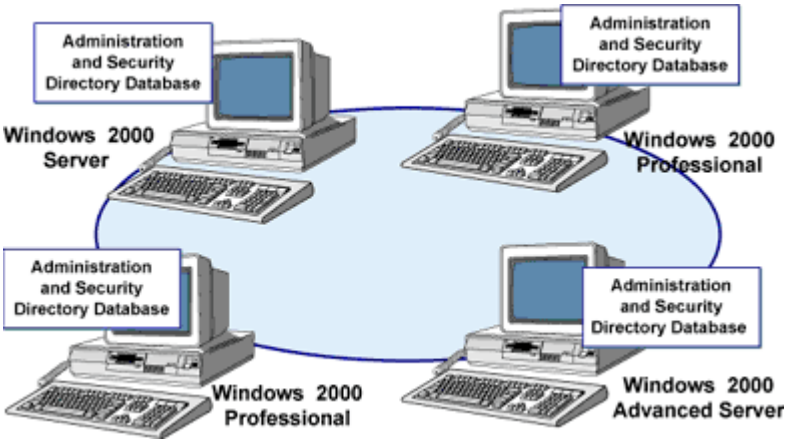
- Processore: *Intel Pentium* 166 (raccomandato uno o più *Pentium* III 300).
- Memoria: 64 MB (raccomandato 128 fino ad un massimo di 8 GB).
- CD-ROM o DVD-ROM *drive* (El Torito-compatible).
- 2 GB di spazio su dischi Ultra IDE (raccomandato Ultra Wide SCSI).
- Scheda di rete (raccomandato *Fast Ethernet*).
- Scheda Video VGA Standard o superiore.
- Tastiera e *Mouse*.
- Per l'implementazione del *cluster* è richiesto *hardware* dedicato.

Microsoft Windows 2000 *Datacenter* è la soluzione *server* ottimizzata per *data warehouse*, *online transaction processing* (OLTP), simulazioni in *real-time*, *Web hosting*.
Microsoft Windows 2000 *Datacenter* ha tutte le caratteristiche di *Microsoft Windows* 2000 *Advanced*

Server, ma supporta fino a 64 GB di memoria RAM e fino a 32 processori SMP.

Per concludere, escludendo la funzionalità di *Clustering*, l'unica differenza tra *Microsoft Windows* 2000 *Server* e le versioni *Advanced Server* e *Datacenter* e nel numero di processori e nella quantità di memoria RAM supportata (scalabilità).

Caratteristiche di Windows2000 - Workgroup



Microsoft Windows 2000 permette di implementare due ambienti di rete in cui gli utenti possano condividere risorse (*file*, stampanti, applicazioni) indipendentemente dalle dimensioni della rete: i *workgroup* ed i domini.

Un *Workgroup Microsoft Windows* 2000 è un insieme logico di *computer*, cominicanti tra loro e che condividono risorse.

Un *workgroup* è chiamato anche rete *Peer-to-Peer* (paritetica) per evidenziare quella che è la sua caratteristica saliente: tutti i *computer* che appartengono ad un *workgroup* sono uguali, senza che ci sia un *server* dedicato alla gestione della sicurezza.

Ogni *computer* che esegua sia *Microsoft Windows* 2000 *Professional* sia *Microsoft Windows* 2000 *Server* gestisce un proprio *security database* locale, cioè una lista di utenti ed impostazioni di sicurezza inerenti il *computer* che ospita tale *database*.

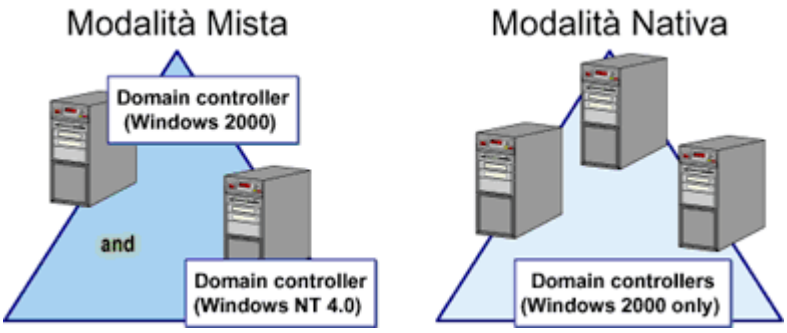
Dunque in un *workgroup* la gestione degli utenti e della sicurezza è decentralizzata, per cui:

- L'utente deve avere un *account* per ogni *computer* che intende utilizzare.
- Se si vuole che l'utente possa utilizzare tutte le macchine ed accedere alle risorse di tutte le macchine, bisogna definire un *account* per ogni *computer*.

Un *workgroup Microsoft Windows* 2000 ha i seguenti vantaggi:

- Non richiede una gestione centralizzata, dunque un *server* dedicato, degli *account* e della sicurezza.
- Non richiede nessuno sforzo particolare relativamente alla progettazione ed all'implementazione.
- È la soluzione più conveniente per ambienti costituiti da pochi *computer* distribuiti su un'area molto limitata (meno di 10 *computer*!).

Caratteristiche di Windows2000 - Domini



Sono in sintesi, gli aspetti essenziali del dominio *Windows* 2000. Come un *Workgroup*, un Dominio *Microsoft Windows* 2000 è un insieme logico di *computer*, cominicanti tra loro e che condividono risorse.

La differenza consiste nel fatto che tali *computer* condividono un *directory database* centralizzato, cioè un *database* che contiene la definizione degli *user account*, dei gruppi e tutte le impostazioni inerenti la sicurezza. Tale *database* è chiamato *Directory* ed è una parte di *Active Directory* che è il *directory services* di *Windows* 2000.

Tale *database* è contenuto su un *server* particolare denominato ***Domain Controller***.

Tale soluzione è affetta da quello che è il tipico problema di tutte le soluzioni centralizzate, il *Single Point of Failure*: in caso di fallimento del *Domain Controller*, le sue funzionalità, tra cui il *database*, vengono a mancare e dunque il dominio cessa di esistere. Per ovviare a questo problema in un dominio è dunque prevista (e consigliata!) la presenza di almeno due *Domain Controller* che dunque garantiscono *Fault Tolerance* e, a regime, bilanciamento dei carichi di lavoro. Infatti tali macchine sono assolutamente paritetiche, entrambe hanno una copia del *Directory* in lettura/scrittura e tali copie vengono periodicamente allineate tramite un processo di replica. Tale ambiente di replica è detto *Multi Master*.

In *Microsoft Windows* NT 4.0 valgono le stesse considerazioni, con la fondamentale differenza che solo uno dei *domain controller*, il *Primary Domain Controller* (PDC), ha il *Directory* in lettura/scrittura mentre tutti gli altri ne hanno una copia in sola lettura e perciò sono denominati *Backup Domain Controller* (BDC). Tale ambiente di replica è detto perciò *Single Master*.

I vantaggi derivanti da un dominio sono:

- Ammnistrazione Centralizzata: da un unico punto amministro tutti gli utenti, tutti i gruppi e la sicurezza.
- *One User One Account*: con un unico *username* ed un'unica *password* l'utente accede al dominio da qualsiasi postazione di lavoro.
- Accesso Universale alle Risorse: una volta validato dal dominio l'utente può accedere a tutte le risorse, indipendentemente dalla loro localizzazione, previa autorizzazione.
- Scalabilità.

Un dominio *Microsoft Windows* 2000 è dunque formato dai seguenti tipi di *computer*:

- *Domain Controllers*: ogni *domain controller* ha una copia in lettura/scrittura del *Directory*, pertanto su ogni *Domain Controller* è possibile creare amministrare e gestire utenti, gruppi, impostare la sicurezza ed ogni *Domain Controller* può effettuare l'attività di validazione. È previsto un meccanismo di replica tra le varie copie del *Directory*.
- *Member Servers*: *computer* che eseguono una qualsiasi versione di *Microsoft Windows* 2000 *Server*, appartengono al dominio ma non svolgono funzionalità di *Domain Controller*.
- *Client Computers*: *computer* che appartengono al Dominio ed eseguono un qualsiasi sistema

operativo *Microsoft*, con ovvia preferenza per *Microsoft Windows* 2000 *Professional* per avere garanzie di stabilità, sicurezza ed affidabilità.

Concludiamo con qualche osservazione relativa alle situazioni di convivenza tra *Domain Controller Windows* 2000 e *Domain Controller Windows* NT 4.0, situazione tipicamente riscontrabile nei casi di *upgrade* a *Windows* 2000.

Per favorire tale convivenza un dominio *Microsoft Windows* 2000 nasce sempre in Modalità Mista, che sostanzialmente disabilita tutte le nuove funzionalità di *Microsoft Windows* 2000 che non sarebbero capite dai *Domain Controller Microsoft Windows* NT 4.0. Non appena questi ultimi sono stati tutti aggiornati al nuovo sistema operativo il dominio può finalmente essere passato in Modalità Nativa che abilita tutte le nuove funzionalità di *Windows* 2000 ma non permette la presenza di *Domain Controller Windows* NT 4.0.

Attenzione poiché il processo di conversione dalla Modalità Mista alla Modalità Nativa è irreversibile, cioè si torna indietro solo installando nuovamente il dominio.

Caratteristiche di Windows2000 - Concetto di Active Directory

Active Directory è il *Directory Services* di *Microsoft Windows* 2000.

Il *Directory Service* è un servizio di rete che ha lo scopo di gestire tutte le informazioni inerenti le risorse di rete per renderle accessibili agli utenti ed alle applicazioni; permette di identificare, descrivere, localizzare, accedere, gestire e rendere sicure tali risorse.

Dunque *Active Directory* fornisce le funzionalità per organizzare, gestire e controllare in maniera centralizzata l'accesso alle risorse di rete, in maniera trasparente rispetto alla topologia di rete ed al protocollo utilizzato. Tramite *Active Directory* è possibile memorizzare ed organizzare un numero praticamente illimitato di oggetti.

Caratteristiche di Linux - Le distribuzioni di Linux

Comunemente col termine *Linux* si intende un sistema completo, composto dal *kernel* (nucleo) più il *software* applicativo, anche se in realtà sarebbe corretto riservare tale denominazione solo per indicare il *kernel* e invece riferirsi al sistema completo come distribuzione di *Linux*.

Al contrario di altri sistemi operativi, in cui il nucleo e i programmi di base vengono realizzati (o forniti in licenza) da un unico produttore, una distribuzione di *Linux* viene realizzata mettendo assieme *software* derivati da progetti e da fonti diverse. Esistono inoltre diverse distribuzioni, realizzate da diversi produttori, che differiscono fra loro in modo anche significativo, sia per i metodi di installazione e configurazione sia per i pacchetti *software* contenuti.

Si deve porre attenzione al fatto che la numerazione delle versioni delle distribuzioni di norma non coincide con quella della versione del *kernel* di *Linux* utilizzata. Ad esempio la versione *Linux Red Hat* 6.1 utilizza un *kernel* versione 2.2.12.

Caratteristiche di Linux - Le licenze Open Source

Il *kernel* di *Linux*, la maggior parte del *software* di sistema ed i programmi più importanti derivano da progetti disponibili sotto una licenza di tipo *Open Source* (a sorgente aperto), delle quali la più utilizzata è la *GNU General Public License* (GPL).

Una licenza *Open Source* differisce in modo sostanziale da una licenza di tipo commerciale, in quanto invece di nascondere il codice sorgente, permette l'accesso ad esso e ne incoraggia la

modifica.

È permesso di includere tutto o parte di un programma distribuito secondo la licenza GPL all'interno di un proprio progetto, nonchè di ridistribuirlo ad altri, a condizione che non se ne cambi la licenza originaria e che si mantenga evidente il *copyright* degli autori precedenti. Una licenza infatti non influisce sulla proprietà intellettuale di un'opera di ingegno, bensì solo sulle modalità con cui l'autore ne concede ad altri l'utilizzo. Per maggiori informazioni si può fare riferimento alla *Copyright FAQ*, che descrive la differenza fra i diversi tipi di licenza:

(ftp://rtfm.mit.edu/pub/usenet/news.answers/law/copyright)

Per i motivi precedentemente elencati, il *software Open Source* viene spesso indicato anche come *software* libero (*free software*), anche se questo termine non deve essere frainteso o generare confusione con il *software* di pubblico dominio (che non ha licenza). La volontà di un autore di distribuire il proprio lavoro assieme ai sorgenti e di renderlo accessibile alla comunità degli sviluppatori viene infatti tutelata dalle licenze *Open Source* con delle limitazioni abbastanza pesanti, la cui non osservanza costituisce la violazione di un documento con valore legale ed è pertanto perseguibile.

Non è ad esempio possibile far diventare un *software Open Source* un prodotto commerciale oppure utilizzarne delle parti all'interno di un prodotto che non sia a sua volta *Open Source*. Questo evita che una *software house* si appropri indebitamente di parti di un programma *Open Source* per avere parte del lavoro già fatto da altri senza dare nulla in cambio alla collettività. Non si tratta comunque di una licenza politica, in quanto le sue finalità sono innanzitutto pratiche. La possibilità di apportare modifiche al *software* o di includerne delle parti all'interno di un proprio programma offre infatti parecchi vantaggi, sia per lo sviluppatore, il quale può semplificare il proprio lavoro utilizzando contributi scritti da altri, sia per l'utilizzatore, il quale, avendo accesso al codice sorgente, può adattare (o farsi adattare) il *software* alle proprie esigenze senza dover sottostare a vincoli e tempi di un unico fornitore.

Le peculiarità del *software Open Source* hanno creato un nuovo modello di sviluppo del *software*, basato sul contributo parallelo allo stesso progetto da parte di una ampia base di programmatori coordinati fra di loro. Questo permette di ridurre notevolmente i tempi di sviluppo e test di un *software*, con una migliore qualità del prodotto finale. Ne è riprova il fatto che i maggiori *software* attualmente disponibili per il mondo *UNIX* (*Linux*, *Apache*, *Sendmail*, Samba, X *Window System*, *Gnome*, KDE, Mozilla, ... solo per citare i più conosciuti), sono sviluppati secondo il metodo *Open Source*.

Caratteristiche di Linux - Libero è diverso da gratuito

Un effetto collaterale delle licenze *Open Source* è che il *software* sviluppato secondo esse è copiabile e distribuibile liberamente, nonchè installabile su un numero qualunque di macchine, in maniera assolutamente legale e senza necessità di acquistare più copie dello stesso prodotto. Questo non significa che ad esempio il produttore di una distribuzione di *Linux* non possa farsi pagare l'eventuale valore aggiunto, come le procedure di installazione, il *packaging* o eventuali programmi commerciali distribuiti assieme al prodotto.

La gratuità del *software*, pur essendo un indubbio vantaggio, non dovrebbe essere considerata la discriminante più importante fra un prodotto *Open Source* ed uno commerciale. Le licenze *Open Source* stanno infatti cambiando il mercato del *software* verso un modello in cui il guadagno non deriva più dal vendere licenze, bensì dal fornire supporto, servizi e professionalità.

Caratteristiche di Linux - Avvertenze e riferimenti

Il fatto che la maggior parte del *software* disponibile su *Linux* sia coperto dalla licenza *GNU* GPL non significa che questa sia una regola applicabile a tutti i programmi. Essa non è infatti l'unica licenza di tipo *Open Source* (che è un termine generale per indicare del *software* il cui codice sorgente sia visibile e modificabile), ma ne esistono altre con vincoli diversi.

Alcuni distributori inoltre sono soliti introdurre nelle proprie versioni di *Linux* anche del *software* commerciale, che ovviamente non gode dei vantaggi di quello *Open Source*, in particolare non è possibile ridistribuirlo. Data la varietà del *software* disponibile su una distribuzione di *Linux*, è sempre buona norma prima di compiere operazioni importanti (ad esempio copiarlo o incorporarlo in lavori derivati) verificarne attentamente caso per caso i termini di licenza.

Networking in Linux - Introduzione

Linux è compatibile con una varietà di *hardware* e supporta differenti protocolli e funzionalità di rete.

Per maggiori informazioni si possono consultare i seguenti documenti del *Linux Documentation Project* di interesse generale, nonché quelli che verranno consigliati trattando i singoli argomenti.

- Autori Vari, *Hardware-HOWTO*
- Autori Vari, *Ethernet-HOWTO*
- Autori Vari, *NET3-4-HOWTO*
- *O.Kirk e T.Dawson*, *The Linux System Administrator Guide, Second edition*

Le Guide di *Linux* e gli *HOWTO* sono disponibili sul sito del *Linux Documentation Project*.

Networking in Linux - Protocolli di rete

TCP/IP

Il supporto per lo *stack* di protocolli TCP/IP è presente fin dalle prime versioni del sistema operativo. Quella offerta da *Linux* è probabilmente una delle implementazioni più affidabili e versatili ed è uno degli elementi che hanno decretato il successo di questo sistema operativo. Dalle versioni 2.2 del *kernel* è supportata anche la famiglia di protocolli TCP/IPv6. Per informazioni a riguardo si faccia riferimento al documento *Linux* IPv6 *HOWTO*.

Linux supporta collegamenti punto-a-punto mediante i protocolli **PPP** (*Point-to-Point-Protocol*), SLIP (*Serial Line IP*) e PLIP (*Parallel Line IP*), il quale consente il collegamento fra due *computer* utilizzando la porta parallela, con velocità fino a 20kB/s. Per maggiori informazioni si consulti il documento PPP *HOWTO*.

IPX/SPX/NCP

È il protocollo proprietario utilizzato dai sistemi *Novell NetWare*. *Linux* può condividere *file* e stampanti in una rete *Novell* sia come *client* che come *server*, inoltre sono supportate le funzionalità di *router* e *bridge* IPX ed è possibile interconnettere reti IPX mediante il *tunneling* di pacchetti IPX sopra il protocollo IP.

La versione commerciale di *Linux* Caldera offre supporto commerciale per un *client NetWare* realizzato da *Novell*, il quale, oltre a consentire l'accesso a *fileserver Novell*, offre funzionalità di *NetWare Directory Service* (NDS).

Per maggiori informazioni si faccia riferimento al documento IPX *HOWTO*.

AppleTalk

Netatalk è una implementazione per i sistemi *UNIX* dei protocolli *AppleTalk*, che permette l'integrazione in reti di *computer Apple* e la condivisione di *file* e stampanti secondo il protocollo *AppleShare*. *Linux* 2.4 inoltre offre, compresi nel *kernel*, il supporto per il *filesystem* HFS ed una implementazione del protocollo *AppleTalk* con funzionalità di *routing*.

Per maggiori informazioni sul *software Netatalk* si faccia riferimento ai siti:

http://thehamptons.com/anders/netatalk/
http://www.umich.edu/~rsug/netatalk/

SMB (*NetBIOS*)

I protocolli delle reti *Microsoft* vengono implementati in *UNIX* mediante il programma Samba, che permette la condivisione di dischi e stampanti verso una rete di PC e offre funzioni di *WINS server* e *Primary Domain Controller*.

Nel *kernel* di *Linux* è inoltre compreso il supporto nativo per i *filesystem* condivisi da macchine *Windows* (smbfs). Per maggiori informazioni si consultino il SMB *HOWTO* e il sito http://www.samba.org/.

WAN *Networking*: ISDN, X.25, *Frame-relay*, ATM, ...

Il *kernel* di *Linux* fornisce di serie **Isdn4*Linux***, un completo sottosistema ISDN, compatibile con l'*hardware* e con i protocolli più utilizzati (V.110, V.120, X.75, audio, ...).

Molti produttori offrono prodotti *hardware* utilizzabili per trasformare una macchina *Linux* in un *router* WAN (E1, E3, X.25, *Frame Relay*, ...). Assieme ad essi generalmente viene anche fornito il *software* che implementa i protocolli necessari.

È inoltre disponibile il supporto per **ATM** e per ADSL. Per maggiori informazioni sull'utilizzo di *Linux* come *router* WAN si consulti il sito http://www.secretagent.com/networking/wan.html

Networking in Linux - Funzionalità avanzate di rete

L'ampia disponibilità di funzionalità e protocolli di rete, unita alle possibilità di personalizzazione ed alla capacità di funzionare su *hardware* limitato, fanno di *Linux* un sistema operativo ideale per creare sistemi *embedded* in grado di sostituire efficacemente *router*, *bridge* ed altri apparati di rete.

Sono disponibili diverse mini-distribuzioni progettate appositamente per questo scopo e sufficientemente ridotte da poter essere installate su un *floppy disk* o su una memoria *Flash* di capacità limitata. Fra le distribuzioni di questo tipo disponibili liberamente vale la pena segnalare *Freesco* e *Linux Router Project*.

È possibile utilizzare le funzioni presenti nel *kernel* per realizzare le seguenti funzioni avanzate:

***Routing* avanzato**

Le versioni recenti del *kernel* di *Linux* offrono funzioni avanzate di *routing* (gestione di più tabelle di *routing*, *policy routing* basato sugli indirizzi IP del mittente o del destinatario o sul contenuto dei pacchetti, gestione delle code e *traffic shaping*, ...) e supportano il *routing multicast* mediante il programma `mrouted`. Mediante opportuni *software*, ad esempio Zebra, è inoltre possibile utilizzare i

protocolli di *routing* più diffusi come RIP, OSPF e BGP4.

Bridging

Il *kernel* di *Linux* supporta la funzionalità di *bridge*, che, instradando opportunamente i *frame Ethernet*, consente di fare apparire una rete segmentata come se si trattasse di un'unica rete. Grazie al supporto per l'algoritmo di *spanning tree* IEEE802.1, un *bridge* basato su *Linux* può cooperare con apparati di altre marche per formare una rete *bridged* di dimensioni maggiori. È inoltre possibile, sfruttando le funzioni di *firewall* interne al *kernel*, associare alle funzioni di *bridge* dei filtri basati su indirizzi MAC, IP o IPX.

Per ulteriori informazioni si leggano il *Bridge+Firewall-HOWTO* e il *Bridge-HOWTO*.

***Firewall* e *routing* avanzato**

Il *kernel* di *Linux* offre funzioni avanzate di filtraggio di pacchetti (`ipchains` nella versione 2.2, `iptables` nella versione 2.4), che possono essere utilizzate per creare dei *firewall* di tipo *stateful* oppure per manipolare i pacchetti in transito per il sistema. Alcune delle applicazioni possibili sono le seguenti:

- realizzazione di *firewall stateful* basati sull filtraggio dei pacchetti;
- *traffic shaping* e gestione avanzata della priorità nel *routing* dei pacchetti;
- modifica del contenuto dei pacchetti in transito (IP *masquerading*, NAT, ...);
- IP *accounting* e statistiche;
- *port forwarding*;
- *transparent proxy*;
- *load balancing*;
- *tunneling* (*mobile IP*, *virtual private networks*);

Oltre alle funzionalità interne al *kernel*, esistono altri pacchetti *software* che implementano funzioni di *firewall* (TIS, SOCKS) o di *proxy* (Squid) per i protocolli più diffusi come HTTP, FTP, telnet, POP3, H.323, ...

Network management

Per *Linux* è disponibile una implementazione del protocollo SNMP (*Simple Network Management Protocol*), che consente il monitoraggio remoto di apparati di rete (*routers*, *bridges*, ...). Per maggiori informazioni si può fare riferimento al sito: http://linas.org/linux/NMS.html.

È inoltre disponibile una vasta gamma di strumenti per la gestione delle problematiche di rete (configurazione, analisi del traffico, ...) e della sicurezza in rete (*intrusion detection*, ...).

Tunneling*, *mobile IP* e *virtual private networks

Il *kernel* di *Linux* permette il *tunneling*, ovvero l'incapsulamento di un protocollo di rete all'interno di un altro. Ad esempio è possibile collegare fra loro via Internet due reti IPX mediante un tunnel di pacchetti IPX all'interno di pacchetti IP.

Il *tunneling* di pacchetti IP all'interno di altri pacchetti IP, permette ad esempio di collegare fra loro due reti in una VPN o di effettuare il *routing* di pacchetti *multicast* su reti non predisposte per tale metdo di indirizzamento. Inoltre facilita l'utilizzo di un *computer* portatile in *roaming* su un'altra rete (*mobile IP*).

Per la realizzazione di VPN, oltre alle semplici funzioni di *tunneling* presenti nel *kernel*, è possibile

utilizzare anche i protocolli standard IPSEC e PPTP (*Point-to-Point Tunneling Protocol*). Per quest'ultimo sono disponibili sia un *client* PPTP, che il programma PoPToP, che implementa la parte *server*, PopToP.

Per maggiori dettagli si faccia riferimento al **VPN mini-*HOWTO***.

Networking in Linux - Interfacce di rete TCP/IP

Nei sistemi *UNIX*, con il termine interfaccia di rete viene indicata una entità logica in grado di scambiare traffico TCP/IP a cui è assegnato un indirizzo di rete, che viene utilizzato come *source address* per tutti i pacchetti IP uscenti da tale interfaccia, a cui può o meno essere associato un corrispondente dispositivo fisico.

Una macchina non ha pertanto un unico indirizzo di rete, bensì potenzialmente tanti quante sono le interfacce di rete attive, o anche un numero maggiore, essendo possibile assegnare alla stessa interfaccia più indirizzi IP.

Un esempio di interfaccia di rete a cui non è associato un dispositivo fisico è l'interfaccia virtuale lo, che permette il collegamento in *loopback* alla macchina locale (ad essa viene generalmente assegnato l'indirizzo standard 127.0.0.1, che prende il nome di indirizzo di *loopback*).

Ad ogni tipologia di interfaccia di rete viene assegnato dal *kernel* un nome simbolico, ad esempio eth per le interfacce associate alle schede *Ethernet*. Se nel sistema sono presenti più interfacce del medesimo tipo, esse vengono numerate in ordine progressivo partendo da zero. Avremo pertanto eth0, eth1, eth2, ... eccetera, in cui la numerazione progressiva avviene in base al *MAC address* delle schede *hardware*.

Alcune interfacce di rete comunemente usate sono le seguenti:

- *loopback*: lo;
- *ethernet*: eth;
- associate a collegamenti PPP: ppp;
- ISDN: isdn;
- associate a collegamenti PPP sincroni su ISDN: ippp.

Si noti che, al contrario degli altri dispositivi, il nome delle interfacce di rete viene creato direttamente dal *kernel* e non esiste pertanto un corrispondente *file* speciale in `/dev`.

SAMBA

Tra le funzioni dei principali sistemi operativi di rete la *suite software* SAMBA implementa in *UNIX* il protocollo SMB (*Server Message Block*). SMB è un protocollo utilizzato per condividere risorse *hardware* (*file*, stampanti, perifieriche in genere) e risorse *software* di un sistema di elaborazione (*pipe*, *mailbox*, eccetera), ed è il più diffuso nell'integrazione di un sistema *UNIX* all'interno di una rete in tecnologia *Windows*.

L'idea del protocollo SMB nasce in ambito IBM verso la metà degli anni '80 e successivamente è stata ripresa da *Microsoft* fin dal 1987. Il protocollo è stato successivamente sviluppato ulteriormente da *Microsoft* e altri.

I dati SMB possono essere incapsulati e trasportati in datagrammi TCP/IP, oppure NetBEUI e IPX/SPX. Nello schema grafico, si evidenzia la posizione del protocollo SMB nell'ambito dell'architettura di comunicazione.

Livelli	OSI	SAMBA				TCP/IP
Livello 7	Application	SMB				
Livello 6	Presentation					Application
Livello 5	Session	NetBIOS		NetBIOS	NetBIOS	
Livello 4	Transport	IPX	NetBEUI	DECNet	TCP/UDP	TCP/UDP
Livello 3	Network				IP	IP
Livello 2	Link	802.2, 802.3, 802.5	802.2, 802.3, 802.5	Ethernet V2	Ethernet V2	Ethernet or other
Livello 1	Physical					

Il boot di un sistema Linux - Device drivers ed organizzazione modulare del kernel di Linux

In *Linux*, oltre ai *driver* delle periferiche, sono state rese modulari e caricabili a *run time* quasi tutte le funzionalità aggiuntive del *kernel*, come il supporto per i *filesystem* o per le funzioni di rete. La strutturazione per moduli permette di ridurre al minimo la dimensione del *kernel* e di caricare in memoria solamente le funzionalità necessarie in un dato momento.

Molti moduli vengono forniti di serie assieme al *kernel* di *Linux* nella *directory* `/lib/modules/versione_del_kernel/`, mentre altri possono essere forniti da terze parti.

Un modulo non è altro che un segmento di codice oggetto (suffisso .o) che per divenire a tutti gli effetti parte del codice del *kernel* deve essere opportunamente linkato ad esso. Tale operazione viene generalmente effettuata mediante i comandi `insmod` o `modprobe`.

La differenza fra `insmod` e `modprobe` consiste nel fatto che il secondo gestisce automaticamente le dipendenze fra i moduli. Infatti il funzionamento di alcuni moduli può dipendere dalla disponibilità di altre funzioni, che pertanto devono essere già caricate in memoria. Ad esempio il *driver* per la scheda di rete NE2000 (ne.o) per funzionare necessita che sia stato prima caricato in memoria il supporto per il *chip* 8390 (8390.o):

```
# insmod /lib/modules/2.2.12-20/net/ne.o
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_open
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ethdev_init
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_interrupt
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol NS8390_init
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_close
```

È pertanto necessario caricare i due moduli nella sequenza corretta:

```
# insmod /lib/modules/2.2.12-20/net/8390.o
# insmod /lib/modules/2.2.12-20/net/ne.o
```

Utilizzando `modprobe` il caricamento dei moduli necessari viene automatizzato, utilizzando una tabella contenente le dipendenze fra i diversi moduli, `modules.dep`, che viene creata dal comando `depmod -a` in uno degli *script* di avvio del sistema.

```
# modprobe ne.o
```

I messaggi di errore o di *logging* generati da un modulo possono essere letti mediante il comando `dmesg`:

```
# dmesg
...
3c59x.c:v0.99H 11/17/98 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html
eth0: 3Com 3c900 Boomerang 10Mbps Combo at 0x6400,
00:60:97:ac:34:a4, IRQ 12
8K word-wide RAM 3:5 Rx:Tx split, autoselect/10baseT interface.
Enabling bus-master transmits and whole-frame receives.
```

Nonostante la maggior parte dei *driver* sia ormai in grado di autoconfigurarsi, spesso risulta necessario passare ad un modulo dei parametri aggiuntivi. Questi possono essere specificati nella linea di comando di `insmod`:

```
# modprobe /lib/modules/2.2.12-20/net/ne.o io=0x300 irq=5
```

I parametri accettati dai *driver* per le diverse schede di rete sono descritti nel *file* `Documentation/networking/net-modules.txt` presente all'interno dei sorgenti del *kernel* di *Linux*.

Il boot di un sistema Linux - Caricamento automatico dei moduli

Linux offre anche la possibilità di caricare un *driver* o una funzione aggiuntiva solo al momento del bisogno ed in modo automatico. È inoltre possibile fare in modo che il modulo venga scaricato dalla memoria dopo un determinato periodo di non utilizzo.

Questa funzionalità viene gestita direttamente dal *kernel*, attraverso il programma `kmod`, oppure nelle vecchie versioni di *Linux* mediante il demone `kerneld`.

Così facendo non è più necessario caricare esplicitamente i moduli necessari, ma è sufficiente associare ad ogni funzionalità il modulo che la gestisca, attraverso il *file* `/etc/modules.conf`. Ad esempio, volendo fare in modo che quando si attiva l'interfaccia di rete eth0 venga automaticamente caricato il modulo ne.o, è sufficiente inserire in `modules.conf` le seguenti linee:

```
alias eth0 ne
options ne io=0x300 irq=5
```

Attraverso `kmod` è possibile gestire le funzioni di caricamento dei moduli in modo complesso, ad esempio eseguendo automaticamente degli *script*. Un esempio più complesso di `modules.conf` è il seguente:

```
alias eth0 8139too
alias parport_lowlevel parport_pc
alias usb-controller usb-uhci

alias char-major-81 bttv
pre-install bttv modprobe -k tuner
options tuner type=0 debug=1
options bttv pll=1 radio=0 card=39
alias sound-slot-0 via82cxxx_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L >/dev/null 2>&1
|| :
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null 2>&1
|| :
```

Per maggiori informazioni è possibile leggere il *file* `Documentation/modules.txt`, presente all'interno dei sorgenti del *kernel* di *Linux*. I parametri accettati dai *driver* per le diverse schede di rete sono descritti nel *file*:

```
/usr/src/linux/Documentation/networking/net-modules.txt
```

Installazione e configurazione di driver per periferiche di rete

Per quanto riguarda *Windows*2000, come per tutti i sistemi *Microsoft*, l'installazione dei *driver* per le periferiche di rete è abbastanza semplice e occorre semplicemente eseguire il programma di installazione fornito in dotazione con la scheda di rete stessa. Se non si disponesse del *driver* relativo o se lo si ritenesse obsoleto di solito si contatta il sito del produttore dell'*hardware*.

Per quanto riguarda i sistemi *Linux* è necessario caricare nel *kernel* l'opportuno modulo contenente il *driver*. Alcuni *kernel* hanno già compilato all'interno i principali *driver* di rete cosicché siano disponibili al momento dell'installazione, in questo caso non occorre fare nulla e all'avvio del sistema sarà disponibile il dispositivo relativo all'*hardware* installato.

In caso contrario ogni specifico *hardware* ha un proprio modulo incluso nella distribuzione dei sorgenti del *kernel*. È bene consultare gli **HOW-TO** relativi al *driver* in questione per poter fornire le opzioni specifiche al momento del caricamento.

Dopo aver caricato il modulo opportuno il dispositivo è utilizzabile e pronto per essere configurato.

Interfacciamento con la rete - Configurazione di una interfaccia di rete

Il comando generalmente utilizzato negli *script* di avvio per configurare ed attivare una interfaccia di rete è:

```
# ifconfig
```

Non sempre il suo utilizzo è obbligatorio: ad esempio le interfacce di rete associate ai link PPP (ppp0, ppp1, ...) vengono attivate direttamente dal demone `pppd` che sovrintende alla connessione.

Se il comando `ifconfig` viene usato senza parametri, permette di ottenere lo stato delle diverse interfacce attive sulla macchina:

```
# ifconfig
eth0 Link encap:Ethernet HWaddr 00:60:97:AC:34:A4
inet addr:193.43.98.1 Bcast:193.43.98.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:440578 errors:0 dropped:0 overruns:0 frame:0
TX packets:2363317 errors:0 dropped:0 overruns:0 carrier:0
collisions:503318 txqueuelen:100
Interrupt:12 Base address:0x6400
ppp0 Link encap:Point-to-Point Protocol
inet addr:213.255.24.25 P-t-P:213.255.6.251 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MTU:1500 Metric:1
RX packets:93509 errors:0 dropped:0 overruns:0 frame:0
TX packets:101886 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:30
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:3924 Metric:1
RX packets:122056 errors:0 dropped:0 overruns:0 frame:0
TX packets:122056 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:0
```

In questo caso nel sistema sono presenti l'interfaccia di *loopback* (lo), una scheda *Ethernet* (eth0) ed un collegamento PPP (ppp0). I valori presenti nell'*output* di `ifconfig` hanno il seguente significato:

- *Link encap*: il tipo di interfaccia (*Ethernet* denota che si tratta di una interfaccia *Ethernet*, mentre *Point-to-Point Protocol* che si sta utilizzando il PPP).
- *HWaddr*: nel caso di schede *Ethernet*, contiene l'indirizzo MAC della scheda.
- *Inet addr*: l'indirizzo IP associato alla interfaccia. Nel caso di collegamento punto-a-punto viene indicato anche l'indirizzo presentato all'altro capo della connessione (P-t-P).
- *Bcast*: l'indirizzo di *broadcast* (ha senso solo nel caso di interfacce col parametro *BROADCAST* attivo).
- *Mask*: la *netmask* associata alla interfaccia (nel caso di collegamenti punto-a-punto fra due macchine, essa vale 255.255.255.255).
- *Flags*: le caratteristiche dell'interfaccia e lo stato della stessa. Nell'esempio, l'interfaccia eth0 risulta attualmente attiva (*UP RUNNING*), collegata ad un medium capace di trasmettere pacchetti IP in *broadcast* (*BROADCAST*) ed in *multicast* (*MULTICAST*) e con dimensione massima dei pacchetti trasmissibili (MTU) pari a 1500 *bytes*. Il parametro NOARP nel caso della interfaccia Punto-a-punto indica che su questa interfaccia non viene gestito il protocollo ARP. Il campo *Metric* viene infine utilizzato per definire la priorità della interfaccia nelle decisioni di *routing*.
- Statistiche: le ultime linee indicano delle statistiche sui pacchetti (*frame*) trasmessi e ricevuti e su eventuali errori.

A proposito del campo *flags* è doveroso fare due brevi richiami:

Le interfacce *ethernet*, oltre a ricevere i pacchetti IP destinati al proprio indirizzo (*unicast*), ricevono anche i pacchetti con destinatario l'indirizzo di *broadcast* della rete. In questo modo è possibile spedire un pacchetto di dati in modo che venga ricevuto da tutte le macchine presenti su uno stesso segmento di rete.

In aggiunta è possibile configurare l'interfaccia per ricevere anche i pacchetti destinati ad un gruppo chiuso di macchine (*MULTICAST*). Tale funzionalità viene sfruttata ad esempio per realizzare trasmissioni multimediali verso più utenti senza dover realizzare un collegamento dedicato per ogni macchina.

Il parametro MTU indica la massima dimensione di un pacchetto trasmissibile su una determinata interfaccia. Le interfacce *Ethernet* hanno in ogni caso un valore di MTU pari a 1500, imposto dall'*hardware* e derivante dalla dimensione dei *frame Ethernet*. Eventuali pacchetti di dati di dimensioni maggiori della MTU possono essere comunque trasmessi, ma prima devono essere deframmentati ed in seguito riassemblati (tali operazioni vengono gestite automaticamente dallo *stack* IP del sistema operativo).

Per configurare manualmente una interfaccia *Ethernet* si utilizza il comando:

```
ifconfig eth0 <ip_address> altri_parametri ...
```

Questo comando assegna un indirizzo IP all'interfaccia e la attiva. Se non vengono esplicitamente specificati dei valori, per gli altri vengono utilizzati dei valori di *default*. La *netmask* e l'indirizzo di *broadcast* vengono calcolati automaticamente in base alla classe dell'indirizzo. Ad esempio se l'indirizzo appartiene ad una rete di classe C, la *netmask* viene posta a 255.255.255.0.

Un esempio di configurazione della interfaccia eth0 è il seguente:

```
ifconfig eth0 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

Per maggiori dettagli e per le altre opzioni utilizzabili nel comando `ifconfig` si faccia riferimento alla pagina del manuale in linea.

Una volta configurata l'interfaccia è necessario creare un opportuno percorso di instradamento (*route*) verso gli indirizzi della rete raggiungibili mediante essa. Nelle versioni recenti di *Linux* il *kernel* crea automaticamente tale *route* quando si configura l'interfaccia. In caso contrario è possibile farlo a mano mediante il comando `route`:

```
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

Le nuove versioni di *route* sono in grado di calcolare automaticamente eventuali valori non specificati e l'interfaccia da utilizzare nel caso non la si indichi nella linea di comando, pertanto si può semplificare il comando precedente in:

```
route add -net 192.168.1.0
```

Impartendo a mano i comandi di configurazione appena visti, essi non vengono salvati. Occorre pertanto inserirli all'interno di un *file* di inizializzazione del sistema. A meno di situazioni particolari, di solito non è necessario usare esplicitamente il comando `ifconfig` o creare a mano un *file* di avvio, in quanto la sequenza di operazioni necessarie viene svolta al *boot* da uno degli *script* di inizializzazione del sistema (in `/etc/rc.d`). Nelle distribuzioni moderne di *Linux* esistono degli strumenti di configurazione che permettono una gestione semplice della configurazione delle interfacce di rete. Purtroppo i meccanismi usati non sempre sono standard fra le varie distribuzioni di *Linux*.

Interfacciamento con la rete - Indirizzamento con DHCP

In una rete contenente un numero elevato di nodi risulta macchinoso assegnare manualmente gli indirizzi IP a ciascuna macchina. La maggior parte delle distribuzioni di *Linux* offrono la possibilità di configurare una interfaccia di rete utilizzando i protocolli **BOOTP** (*Boot Protocollo*) oppure **DHCP** (*Dynamic Host Configuration Protocol*), che consentono di centralizzare l'assegnazione degli indirizzi attraverso l'uso di un apposito *server*.

Allo scopo viene utilizzato il demone `dhcpcd`, il quale si occupa di richiedere al *server* l'indirizzo. Ad esso è associato un **periodo di validità (*lease*)**, scaduto il quale il demone, che pertanto deve rimanere attivo, si occupa di richiedere un nuovo indirizzo.

Poiché al momento del *boot* la macchina non dispone ancora di un indirizzo IP, la richiesta al *server* viene fatta mediante un *frame Ethernet* spedito in modalità *broadcast*.

Mediante BOOT o DHCPD è possibile ottenere anche altri tipi di informazioni, ad esempio il nome della macchina o del dominio di appartenenza e gli indirizzi dei *server* DNS da utilizzare per la risoluzione dei nomi. I dati ottenuti da `dhchcp` per le diverse interfacce vengono salvati in un *file* `/etc/dhcpc/dhcpcd-<nome_interfaccia>.info`.

Quando `dhcpcd` ottiene dei dati da un *server*, esso esegue lo *script* `/etc/dhcpc/dhcpcd-eth0.exe`, che contiene i comandi necessari a riconfigurare l'interfaccia con il nuovo indirizzo e ad aggiornare il sistema con i valori ricevuti.

Per ulteriori informazioni si faccia riferimento al DHCP Mini *HOWTO*.

Interfacciamento con la rete - IP aliasing

Mediante `ifconfig` è anche possibile assegnare alla stessa interfaccia fisica più indirizzi IP. Tale funzione è supportata ad esempio nel caso delle schede *Ethernet*, per le quali è possibile creare delle interfacce logiche denominate ethN: M (ad esempio: eth0:1, eth0:2, ...) con indirizzi diversi.

Questo permette di gestire servizi virtuali che rispondano ad indirizzi diversi sulla stessa macchina. Ad esempio è possibile configurare un *server Web* in modo che presenti pagine diverse a seconda dell'indirizzo utilizzato per accedervi.

Per configurare un indirizzo IP secondario su una interfaccia *Ethernet* si utilizza il comando `ipconfig` con la seguente sintassi:

```
ifconfig eth0:1 192.168.1.23 netmask 255.255.255.0 up
```

Se non esiste già, è necessario ricordarsi di creare una *route* verso l'indirizzo della nuova interfaccia:

```
route add 192.168.1.23 dev eth0:1
```

Configurare il routing per la gestione dei pacchetti di una rete locale verso Internet

L'operazione di *routing* consiste nelle decisioni che il *kernel* deve intraprendere per selezionare l'interfaccia di rete mediante cui inviare un pacchetto IP destinato ad un determinato indirizzo: ad esempio un pacchetto destinato ad una macchina appartenente allo stesso segmento di rete *ethernet* di una delle interfacce del sistema viene spedito direttamente al destinatario.

Nel caso il destinatario non sia raggiungibile direttamente, il pacchetto di dati può essere dato in consegna ad un *router* che si occupi del suo inoltro. Questo dispone di una interfaccia sulla stessa rete della macchina e di una interfaccia su un'altra rete. In questo modo, passando per uno o più *router*, è possibile far giungere il pacchetto IP alla destinazione desiderata. Se nella rete locale non esiste una *router* attraverso cui è possibile raggiungere la macchina destinataria di un pacchetto, questo viene scartato. Generalmente viene definito un *router* a cui spedire tutti i pacchetti non recapitabili attraverso altre strade, che prende anche il nome di *default gateway*.

Per *default*, nelle versioni del *kernel* di *Linux* superiori alla 2.2, il *forwarding* di pacchetti IP, ovvero il passaggio per una macchina di pacchetti generati da altre macchine e non diretti ad essa, non è abilitato.

Volendo utilizzare il proprio sistema come un *router*, il *forwarding* deve essere attivato in modo esplicito mediante il seguente comando:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Si tratta di un esempio significativo di come sia possibile utilizzare il proc *filesystem* per modificare il funzionamento del *kernel* a *run time*. Altri esempi di *tuning* del *kernel* sono documentati nel *file* `net/TUNABLE` fornito con i sorgenti del *kernel* di *Linux*.

Nei sistemi *Windows* si può abilitare la stessa funzione accedendo a *Start* -> Impostazioni -> Pannello di controllo -> Strumenti di amministrazione -> Servizi -> *Routing* e Accesso remoto. Selezionando le Proprietà con il tasto destro del *mouse* ed abilitando il modo Automatico nella casella Tipo di Avvio.

Per rendere effettive le modifiche apportate, cliccare il pulsante Applica.

Nel caso più banale la decisione sull'instradamento viene effettuata semplicemente confrontando l'indirizzo di destinazione del pacchetto con il contenuto di una tabella di *routing* statico. In reti complesse è possibile utilizzare uno dei protocolli di *routing* dinamico (RIP, OSPF e BGP4, ...) oppure utilizzare le funzioni di filtraggio di pacchetti interne al *kernel*, per decidere i percorsi in base a criteri più complessi, come la porta TCP o UDP sorgente o di destinazione, il tipo di protocollo oppure la qualità di servizio desiderata (QoS).

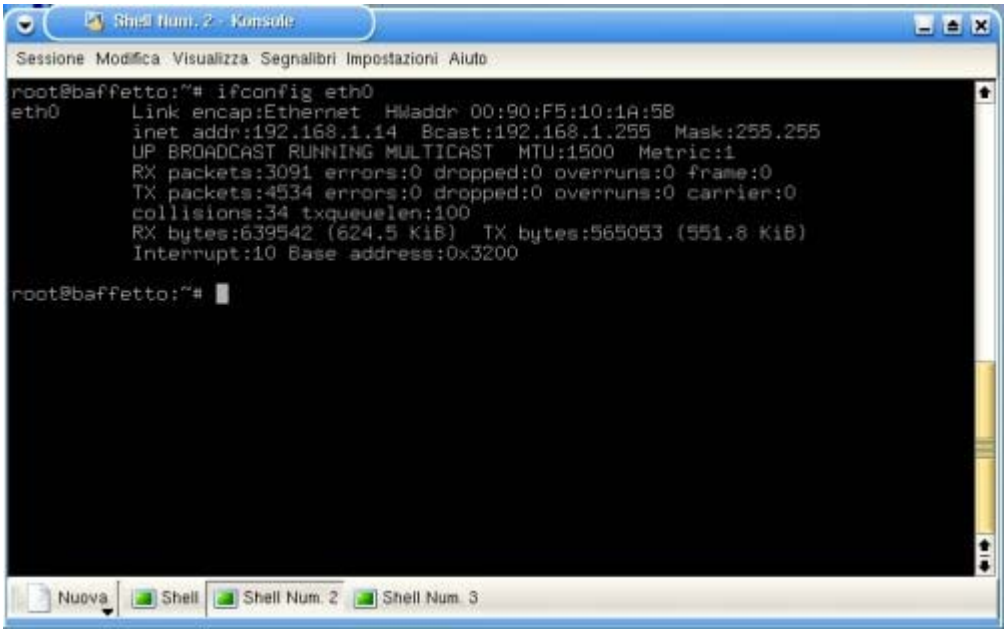
Il comando netstat

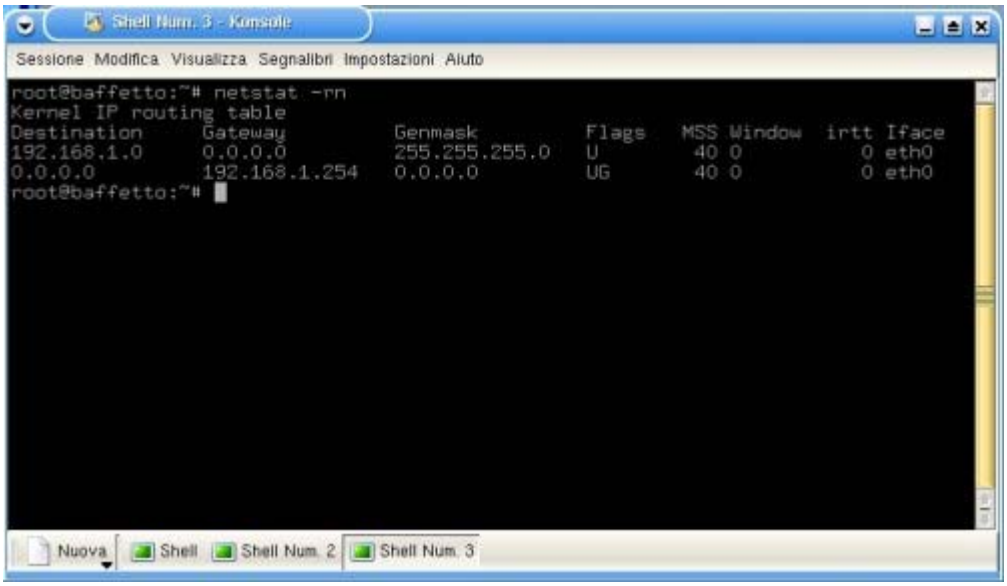
Per visualizzare il contenuto corrente della tabella di *routing* si utilizza per tutti e due gli ambienti operativi il comando `netstat`.

Riportiamo come esempio la tabella di *routing* di una macchina con una interfaccia *Ethernet* sulla rete di classe C 192.168.1.0.
Il suo indirizzo IP è 192.168.1.14 e viene utilizzato come instradamento di *default* per il traffico locale. Nella rete è presente un *router* all'indirizzo 192.168.1.254.

Shell di *Linux*:

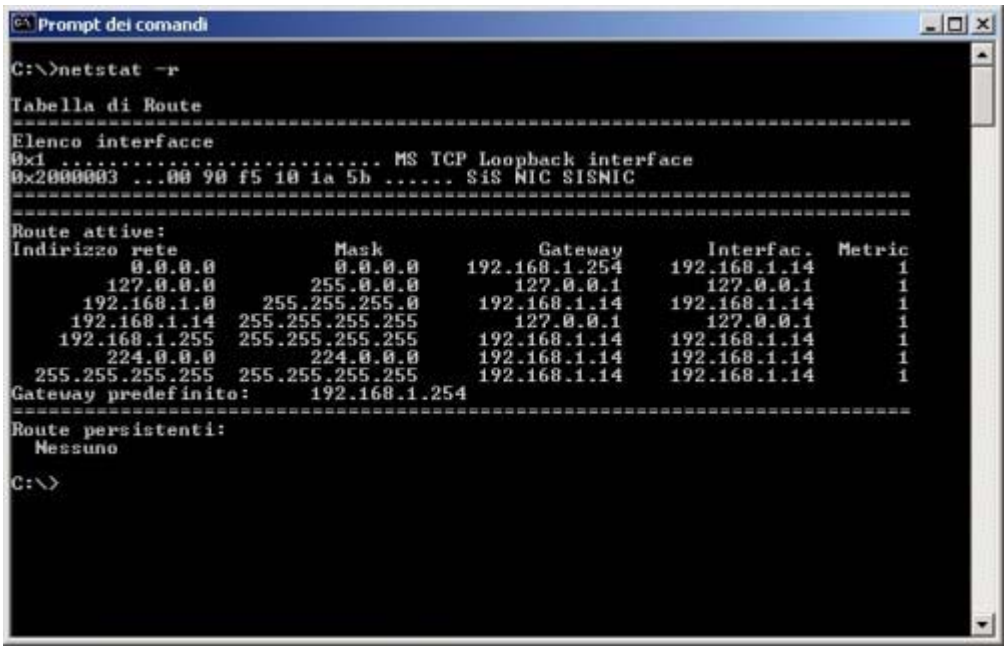
- `netstat -h` fornisce un *help* in linea con la sintassi del comando.
- `netstat -r` stampa il contenuto della tabella di *routing*.
- `netstat -rn` stampa il contenuto della tabella di *routing* senza che gli IP vengano risolti come nomi simbolici.





Prompt di MS-DOS di *Windows*2000:

- `netstat /?` fornisce un *help* in linea con la sintassi del comando.
- `netstat -r` stampa il contenuto della tabella di *routing*.



L'output del comando netstat

I valori significativi che appaiono nell'*output* di `netstat` sono i seguenti:

- *Destination* o Indirizzo rete: identifica la rete di destinazione della *route*. Questo valore per avere significato deve essere letto assieme al valore della *netmask* (*Genmask* o *Mask*).
- *Gateway*: indica l'eventuale *router* da utilizzare per raggiungere la destinazione specificata.

La tabella di *routing* contiene un insieme di percorsi di instradamento (*route*), che possono essere di diverso tipo:

- una *route* di tipo H (*host*) identifica un percorso utilizzato per raggiungere un determinato *host* (*flag* H);
- una *network route* specifica invece il percorso da utilizzare per raggiungere tutte le macchine di una rete specificata (il campo *Genmask* contiene la *netmask* associata alla rete);
- una *default route* specifica quale interfaccia (o *gateway*) utilizzare come ultima risorsa se non sono stati trovati degli altri percorsi validi.

Un instradamento può fare riferimento ad una interfaccia oppure passare attraverso un *gateway* (identificato dal *flag* G).

Per interagire con la tabella, creare nuovi instradamenti o eliminare quelli esistenti, si utilizza il comando *route*.

NOTA: nel caso sopra descritto, non è necessario aggiungere altre *route* statiche, in quanto nella rete è presente un *router* che è in grado di instradare i pacchetti non diretti alla rete locale.

Esempi di utilizzo del comando route nella shell di Linux

Aggiungere una *route* statica per l'instradamento verso la rete 192.168.50.0/24, attraverso l'indirizzo IP 192.168.1.253:

```
route add -net 192.168.50.0 netmask 192.168.50.255 gw 192.168.1.253
```

Cancellare una *route* statica per l'instradamento verso la rete 192.168.50.0:

```
route del -net 192.168.50.0
```

Creare una *route* di *default* utilizzando un *default gateway*:

```
route add default gw 192.168.1.254
```

Cancellare una *route* di *default*:

```
route del default gw 192.168.1.254
```

Le versioni recenti del comando `ifconfig` aggiungono automaticamente delle *route* appropriate per raggiungere le interfacce che vengono configurate, inoltre per rendere permanenti le configurazioni del *routing* di *Linux* occorre modificare gli *script* situati in `/etc/sysconfig/network-scripts/` in *RedHat Linux*.

Esempi di utilizzo del comando route nella finestra prompt di ms-dos di Windows2000

Aggiungere una *route* statica per l'instradamento verso la rete 192.168.50.0/24, attraverso l'indirizzo IP 192.168.1.253:

```
route ADD 192.168.50.0 MASK 255.255.255.0 192.168.1.253
```

Cancellare una *route* statica per l'instradamento verso la rete 192.168.50.0

```
route DELETE 192.168.50.0
```

Creare una *route* di *default* utilizzando un *default gateway*:

```
route ADD 0.0.0.0 MASK 0.0.0.0 192.168.1.254
```

Cancellare una *route* di *default*:

```
route DELETE 0.0.0.0
```

Aspetti gestionali e procedure per gestire più server sulla stessa rete

La parte di gestione di reti e applicazioni è la parte del *networking* che richiede più sforzi per lo sviluppo. Il problema risiede nella poca uniformità delle soluzioni che sono in commercio, dovuto ai diversi approcci utilizzati. Questo rende la gestione uno degli aspetti più delicati e costosi. Per migliorare questa situazione, si sta tentando un approccio di tipo centralizzato, anche se si è lontani da uno standard che possa unificare le soluzioni utilizzate.

In aziende di ogni tipo e dimensione è sempre più essenziale la disponibilità di un efficiente Sistema Informativo Aziendale, la cui dimensione e complessità devono essere ovviamente rapportate alle dimensioni ed alla struttura aziendale.

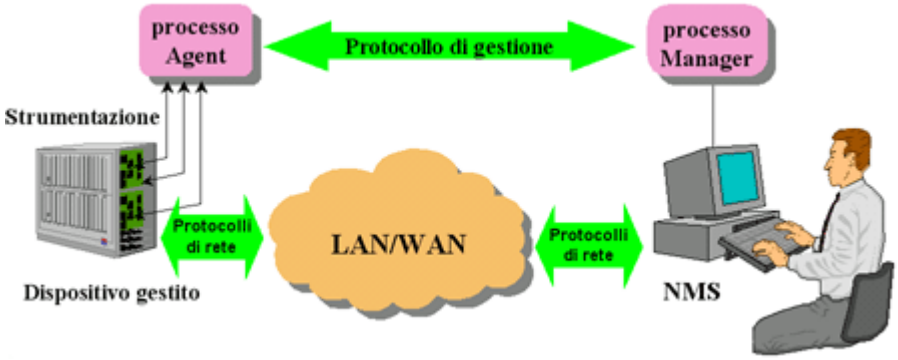
Quando le dimensioni dell'azienda sono rilevanti (centinaia o migliaia di dipendenti), ed a maggior ragione se l'azienda è anche distribuita territorialmente, le infrastrutture di rete utilizzate come supporto ai trasferimenti di dati nell'ambito del sistema informativo diventano realmente molto complesse. Tali infrastrutture sono prevalentemente costituite da un numero sempre crescente di reti locali, o LAN (*Local Area Network*), interconnesse tra loro localmente o per mezzo di collegamenti geografici. I responsabili della gestione di tali reti si trovano pertanto a dover gestire sistemi sempre più complessi con risorse umane sempre più limitate. Una delle strade percorribili per ottenere un

incremento del grado di efficienza consiste nell'utilizzo di tecnologie che consentano il controllo centralizzato delle strutture di rete e dei servizi realizzati per il loro tramite.

Gli obiettivi da raggiungere, legati alla necessità di gestire le applicazioni che stanno alla base delle attività aziendali, sono essenzialmente i seguenti:

- massimizzare il livello di disponibilità della rete;
- contenere al minimo i costi di gestione;
- ottimizzare le prestazioni e la qualità del servizio fornito;
- identificare per tempo le nuove esigenze al fine di pianificare l'evoluzione della rete.

Alla base della gestione di rete c'è quindi un colloquio tra la stazione di gestione e l'apparato gestito. Tale colloquio si esplica in particolare tra due entità, realizzate per mezzo di processi *software*, denominate rispettivamente *Manager*, nel centro di gestione, ed *Agent*, nel nodo gestito. Il trasferimento di informazioni tra *Manager* ed *Agent* avviene in accordo ad un insieme di regole, sintattiche e semantiche, che costituiscono il protocollo di gestione. Il protocollo di gestione è un protocollo di livello applicativo che si appoggia sulla pila protocollare sottostante.



In ambito **OSI** tale protocollo si chiama **CMIP** (*Common Management Information Protocol*) ed in ambito SNMP si chiama appunto **SNMP** (*Simple Network Management Protocol*).

Installare e configurare software antivirus

Per combattere efficacemente la diffusione dei virus in sistemi non dotati di un controllo efficace sui permessi di lettura e scrittura degli utenti, è bene installare *software* che controllino periodicamente i dati e *file* sensibili del sistema.

Questi *software* sono chiamati Antivirus e vengono aggiornati continuamente per fare fronte alle evoluzioni dei virus esistenti ed alla diffusione di quelli nuovi. Prevalentemente vengono installati su macchine *client* con sistemi *Windows*.

L'installazione non presenta alcuna difficoltà, occorre però mantenere il proprio antivirus aggiornato alle ultime versioni onde evitare di contrarre virus recenti non rilevabili da un antivirus obsoleto.

Citiamo alcuni antivirus tra i più conoscuti:

- *Norton* (www.symantec.com)
- *NetShield* (www.mcafeeb2b.com)
- *Protector* (www.pspl.com)