Ministero dell'Istruzione, dell'Università e della Ricerca Servizio Automazione Informatica e Innovazione Tecnologica

Modulo 17

Internet server

ForTIC

Piano Nazionale di Formazione degli Insegnanti sulle Tecnologie dell'Informazione e della Comunicazione

Percorso Formativo C

Materiali didattici a supporto delle attività formative

2002-2004

Promosso da:
- Ministero dell'Istruzione, dell'Università e della Ricerca, Servizio Automazione Informatica e Innovazione Tecnologica
- Ministero dell'Istruzione, dell'Università e della Ricerca, Ufficio Scolastico Regionale della Basilicata

Materiale a cura di:
- Università degli Studi di Bologna, Dipartimento di Scienze dell'Informazione
- Università degli Studi di Bologna, Dipartimento di Elettronica Informatica e Sistemistica

Editing:
- CRIAD - Centro di Ricerche e studi per l'Informatica Applicata alla Didattica

Progetto grafico:
- Campagna Pubblicitaria - Comunicazione creativa

In questa sezione verrà data una breve descrizione del modulo.

Gli scopi del modulo consistono nel mettere in grado di:

- Installare e configurare un *Web server*, un *proxy server*, un *list server*, un *chat server*, un *news group server*.
- Installare e configurare un *firewall* e saperne spiegare necessità e funzioni.

Il modulo è strutturato nei seguenti argomenti:

- **Installazione di *server***
  - Installare e configurare un *Web server*.
  - Installare e configurare un *proxy server*.
  - Installare e configurare un *list server*.
  - Installare e configurare un *chat server*.
  - Installare e configurare un *news group server*.
- ***Firewalls***
  - Spiegare i motivi per cui è necessario un *firewall* e le sue funzioni.
  - Installare e configurare un *firewall*.

Introduzione

Installazione di server

Cosimo Laneve

Struttura di un server Web - Cosa è un Web server

Un ***server*** *Web* è sostanzialmente un *software* che fornisce all'utente pagine **HTML** da visualizzare nel proprio *browser*. Il **protocollo** su cui si basa è detto **HTTP**. Tale **protocollo** stabilisce quali sono le regole per richiedere documenti ad un ***server***. Questo **protocollo** è diviso in due parti: la richiesta del ***client*** e la risposta del ***server***. Il **protocollo** è basato completamente su testo ASCII, ma non stabilisce il formato della risorsa restituita al ***client***. In altre parole è compito del *browser* ricevere il documento (pagina HTML, immagini gif, eccetera) e visualizzarlo correttamente.

Il server HTTP

Un ***server*** HTTP è il programma (anche detto demone) che gestisce e soddisfa le richieste **HTTP** ricevute su di un particolare canale di un elaboratore. In pratica è il programma che realizza le potenzialità di **HTTP** e che ne implementa le caratteristiche.

Questo tipo di programmi rimangono in una situazione di attesa fino a quando non vengono risvegliati dalla presenza di un segnale sulla porta loro assegnata, che in genere è la 80. A questo punto il ***server*** legge il segnale e ne controlla la sintassi (definita dal protocollo **HTTP**). Se la sintassi viene riconosciuta come valida, il demone, interpretata la richiesta, cerca di soddisfarla; sia che il ***server*** riesca ad accedere ai dati richiesti, sia che non ci riesca, comunque esso manda una risposta o contenente i dati richiesti o indicante la situazione di errore.

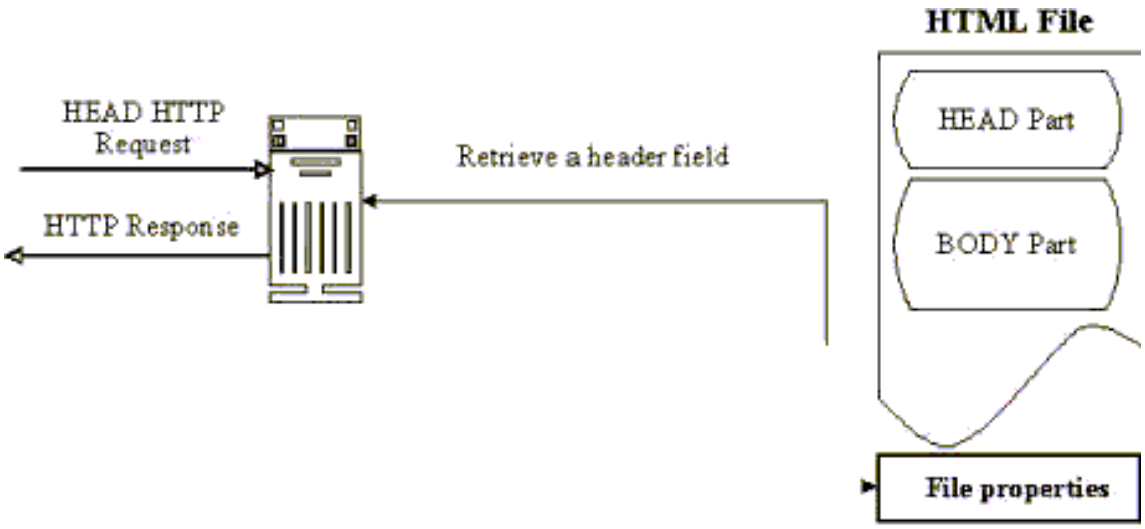
Richiesta di una pagina Web e risposta di un server Web

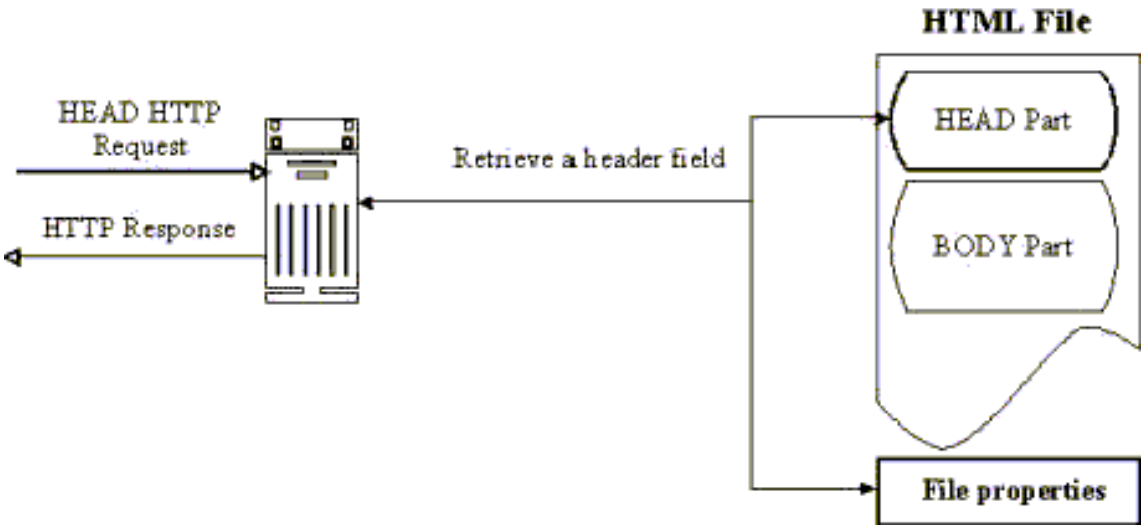
Questo funzionamento non è comunque caratteristico dei soli HTTPd (demoni **HTTP**) ma anche di molti ***server*** che implementano altri protocolli come **FTP** (*File Transfer Protocol*), *Gopher* o NNTP (*Network News Transfer Protocol*). Al ***server*** viene comunque assegnato uno spazio fisico sull'*hard disk* in cui può rintracciare i documenti richiesti, e che viene identificato come spazio del *Web* o Web space. Un documento, quindi, per poter essere distribuito, deve essere contenuto in questo spazio altrimenti il ***server*** non riesce a recuperarlo.

Come succede con la gran parte dei sistemi *client-server*, quando il ***server*** riceve una richiesta si attivano le seguenti attività:

- genera un sub-**processo** per rispondere alla richiesta;
- allo stesso tempo gli altri processi continuano ad ascoltare sulla porta 80 per nuove richiesta;
- il sub-**processo** generato risponde alla richiesta.

Compito del *Web* ***server*** è anche quello di fare da *gateway* tra i dati ricevuti con una richiesta e una **applicazione**. In questa maniera è possibile accedere ad archivi, piuttosto che modificare *run time* un *file* in dipendenza ai dati forniti oppure per poter aggiornare i dati di una pagina **HTML** (pensiamo ad una pagina che mostra il numero di accessi al *Web* ***server***). Alcune implementazioni di HTTPd permettono di cifrare i messaggi, per poter effettuare delle transizioni di dati in maniera sicura (protocollo **SSL**) oppure richiedere al ***client*** un'autenticazione, per poter selezionare gli utenti a cui concedere l'accesso.





Questo **protocollo** è diviso in due parti: la richiesta del ***client*** e la risposta del ***server***. Il **protocollo** è basato completamente su testo ASCII.

Richiesta di un *client*

La richiesta è una linea di testo divisa in 3 parti:

- tipo di richiesta.
- **URL**.
- Protocollo usato.

Le richieste possibili al punto 1 sono: ***GET***, ***POST***, ***HEAD***, *PUT*, *DEL*, *TRACE*. L'**URL** è il percorso al quale si vuole accedere nel chiedere un oggetto, il quale deve essere comprensivo del nome dell'oggetto. Ad esempio:

www.aipa.it/prove/pagina.asp?par=val&par2=val2

- **www.aipa.it** è il **nome di dominio**;
- **/prove/** è la cartella locale del *Web-**server***;
- **pagina.asp** è il documento richiesto;
- **?par=val&par2=val2** rappresenta un elenco di parametri che possono essere inviati al *Web **server***.

Formato dei messaggi HTTP

Formalmente una messaggio **HTTP** 1.0 è composto da una richiesta e una risposta:

Request = Request-line *(General-Header | Request-Header | Entity-Header) CR LF [Entity-Body] Response = Status-Line *(General-Header | Request-Header | Entity-Header) CR LF [Entity-Body] Request-Line = Method (Spc) Requested-URI (Spc) HTTP-Version CRLF General-Header = Date | MIME-Version | Pragma Request-Header = Authorization | From | If-Modified-Since | Referer | User-Agent Entity-Header = Allow | Content-Encoding | Content-Lenght | Content-Type| Expires |Last-Modified | extension-header Entity-Body = *OCTET (qualsiasi sequenza di dati codificati a

Formato dei messaggi HTTP

I Metodi

GET*:** serve per recuperare qualsiasi tipo di *file* (intesa come entità) identificato dall'URI specificato. Un esempio di richiesta **HTTP** di tipo ***GET, e relativa risposta può essere:

GET http://www.aipa.it/index.html HTTP/1.0 HTTP/1.0 200 OK
Date: Fri, 24 Jun 2002 11:33:18 GMT
Server: Apache/1.0.0
Content-type: text/html
Content-length: 3395
<html>
<head>
<title> Sito Web AIPA </title>
</head>
[...]

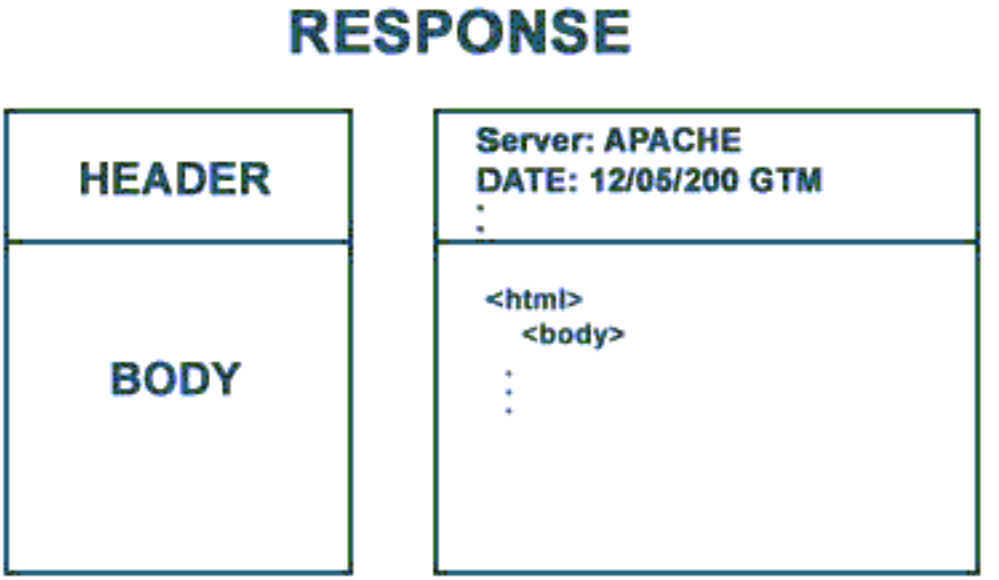
HEAD*:** è un metodo molto simile a quello del ***GET, eccetto che serve a recuperare solo gli attributi di un *file* e non il suo contenuto. È il metodo lanciato dai *browser* per controllare se il *file* richiesto non sia già presente in locale (controllando l'*header*). Se così fosse non sarebbe necessario richiederlo inutilmente al ***server***. In pratica fornisce come risposta gli stessi dati di una richiesta di tipo ***GET*** a parte l'*Entity-Body*. Per effettuare una richiesta di tipo ***HEAD***:

HEAD http://www.aipa.it/index.html HTTP/1.0 HTTP/1.0 302 Found
MIME-Version: 1.0
Server: IIS/4.0
Date: Friday, 23-Jun-02 19:40:14 GMT
Location: http://www.aipa.it/index.html
Content-Type: text/html

Content-Length: 2133
POST*:** questo metodo è usato per richiedere che il ***server di destinazione accetti l'entità acclusa al pacchetto di richiesta come sotto-coordinata della risorsa identificata dall'URI nella *Request-Line*. In pratica viene usato per poter passare dei parametri ad un programma di elaborazione, attivato via richiesta **HTTP**, che spesso manda una risposta dipendente dalla *query*. Questo metodo non obbliga al *Web* ***server*** a fornire una pagina oggetto come risposta, ma solo la testata, per cui è utili nel caso in cui una richiesta non deve modificare la pagina da cui è stata lanciata.

Risposta del *server*

La prima linea parla del **protocollo** usato e restituisce un valore del ***server*** (un valore superiore a 400 indica un errore). È seguito dalla data, la versione del ***server*** e la data dell'ultima modifica dell'**URL** (permette al ***client*** di sapere se i *file* nella sua *cache* sono ancora validi). *Content-Length* è la lunghezza della risposta (richieste a *script* ***CGI*** non forniscono queste informazioni) e *Content-Type* comunica al *Web* ***client*** il tipo di ***MIME*** usato nella risposta (testo, HTML, immagini ...).



Formato dei messaggi di risposta HTTP

Un errore: proviamo a connetterci ad un ***server*** HTTP tramite *telnet*, ed osserviamone la risposta (in italico quello che risponde il sistema).

>telnet www.aipa.it 80 *Trying 195.213.25.18... Connected to www.aipa.it* get / HTTP/1.0 <invio> *HTTP/1.1 501 Method Not Implemented Date: Mon, 27 Sep 1999 21:22:03 GMT Server: Apache/1.3.3 Connection: close*
Come potete notare, l'*header* rende inutile alcuna spiegazione, ed alla fine della connessione verremo buttati fuori col codice di errore 501. L'**HTTP** è un **protocollo** molto semplice, come è ben evidenziato da questi esempi:

>telnet www.iol.it 80 *Trying 195.213.125.118... Connected to www.iol.it* GET / <

invio > *[segue sullo schermo il contenuto della pagina di default del sito iol]*
Cosa è successo dentro il ***server*** *Web*? Vi siete connessi con il *telnet* alla porta 80 di www.iol.it (**indirizzo** *IP* 195.213.125.118) (la 80 è la porta standard del ***server*** HTTP). Il ***server*** è stato scritto per rispondere a delle richieste e voi avete scritto ***GET*** / seguito da *return*.

Codici di errore: riportiamo in seguito i codici di errore che restituisce un ***server*** HTTP nelle varie circostanze.

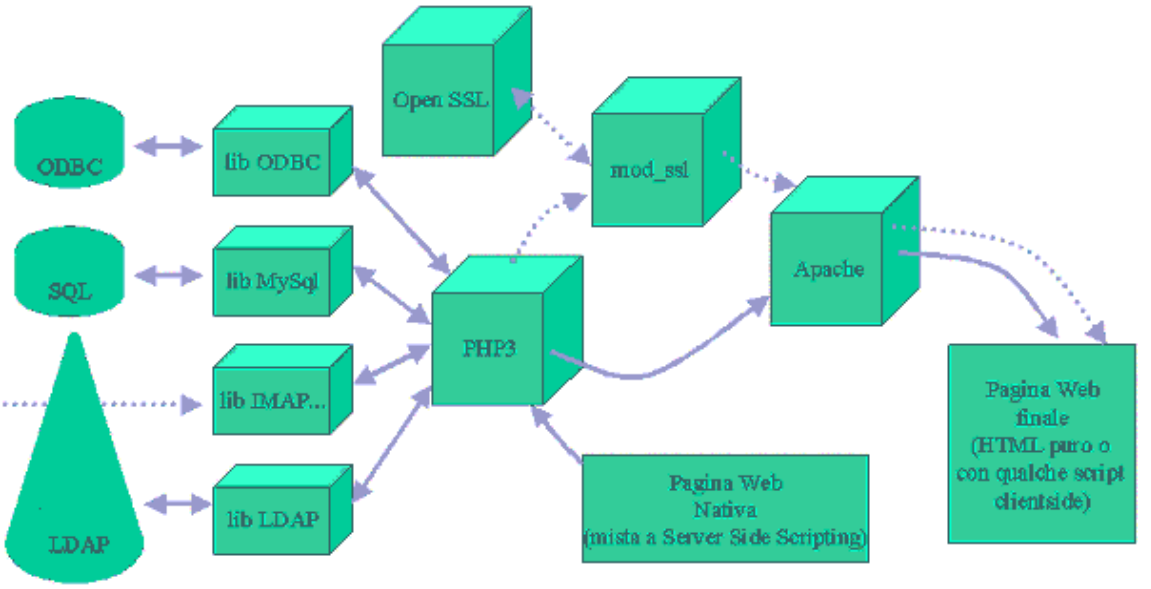
Codice	Descrizione
400	*Syntax Error*
401	Non autorizzato. Richiesta autenticazione
402	*Browser does not support required encryption method. Challenge / Response is being used*
403	Rifiutato. Esempio, cercare di aprire una pagina **SSL** con un *browser* non abilitato
404	*File* non trovato. *Link* errato o spazi nel nome di *directory*
405	*Method Not Allowed*. Problemi con Mime
406	*Not Acceptable*
407	*Proxy Authentication Required*
412	*Precondition Failed*
414	*Request-URI Too Long*. O si è sotto attacco *hacker*, o vi sono problemi col comando ***POST***
500	*Internal Error*. Esempio, lo *user* ***Anonymous*** non ha diritti **ACL** sui *file* che deve accedere
501	*Not Implemented*
502	*Bad Gateway*. Esempio, cercare di utilizzare un IDC con **DNS** sbagliato

Una definizione di ***server*** *Web* siffatta però rischia di essere allo stato attuale riduttiva: i moderni *Web* ***server*** oltre che restituire informazioni sono in grado di comunicare con *Application Server*, *software* in grado di elaborare informazioni, restituite poi al ***client*** sotto forma di pagine **HTML** dette dinamiche. La possibilità elaborativa è divenuta essenziale poiché permette al *Web* ***server*** di funzionare come interfaccia *Web* (e quindi indipendente dalla piattaforma) verso un qualunque sistema informativo. In altri termini un ***client*** chiede elaborazioni remote ad un sistema complesso, che opera su una qualunque piattaforma, veicolando le richieste attraverso una interfaccia semplice, diffusa e generica: il ***server*** *Web*.

Server Web: Apache

Apache (il cui *team* di sviluppo è formato da volontari, noti come *Apache Group*, www.apache.org) è nato come sostituto per il *Web* ***server*** HTTPd 1.3 sviluppato dal

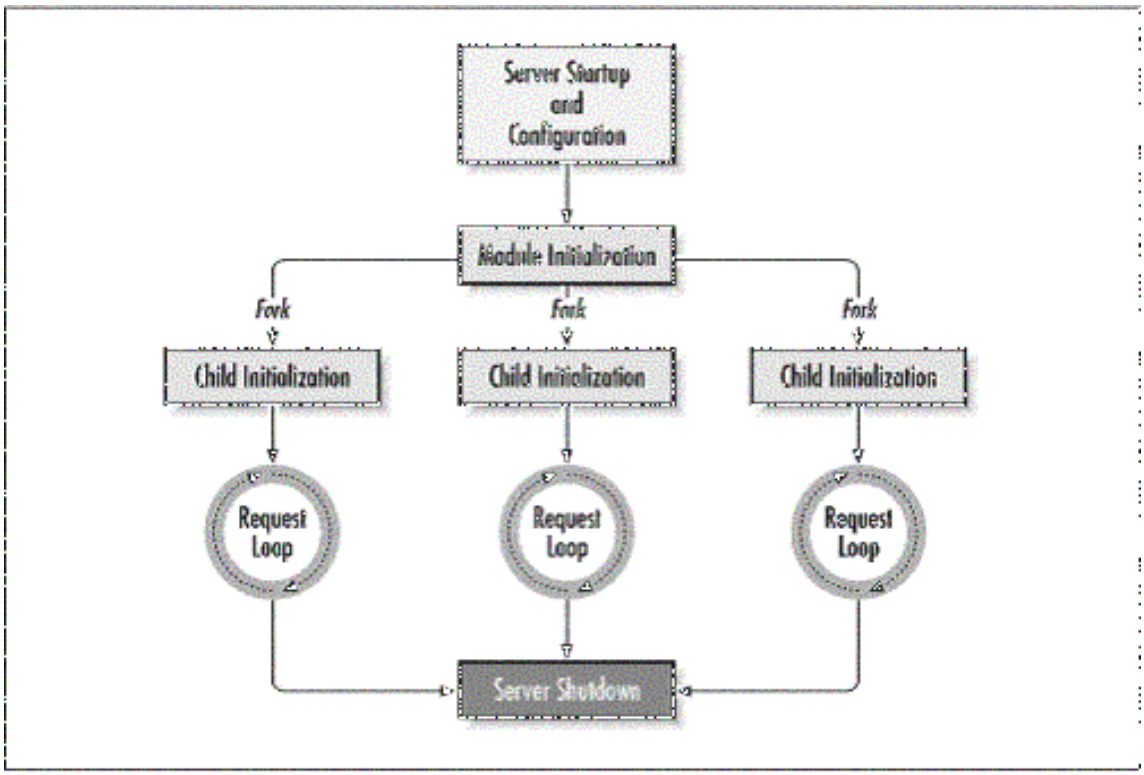
NCSA (*National Center for Supercomputing Applications*), inglobandone le caratteristiche, risolvendone i problemi ed implementando nuove caratteristiche. È il prodotto al momento più diffuso sulla rete, contando il doppio circa delle installazioni del suo diretto concorrente, **IIS** di *Microsoft*.



Le interazioni di Apache con latri moduli

Le ragioni di tale successo vanno attribuite alle sue caratteristiche di flessibilità ed affidabilità, ma ancora di più alla filosofia commerciale che lo contraddistingue: è un prodotto open e liberamente distribuibile, così come il sistema operativo che lo ospita, *Linux*: in definitiva oltre alla flessibilità permette di contenere i costi di gestione, nonostante possiede caratteristiche in tutto e per tutto simili (se non superiori) alla diretta concorrenza.

Apache è un *Web* ***server*** che gestisce il protocollo **HTTP**, gira come un **processo** *stand-alone*, senza cioé chiedere l'appoggio ad altre applicazioni né direttamente all'utente. Per poter fare ciò, ***Apache***, una volta che è stato avviato, crea dei sottoprocessi (comunemente detti processi *children*) per poter gestire le richieste: questi processi, non interferiscono mai con il **processo** padre, ma può succedere l'opposto: quando il **processo** padre viene terminato, ad esempio con un segnale di *stop*, anche i *children* saranno terminati.



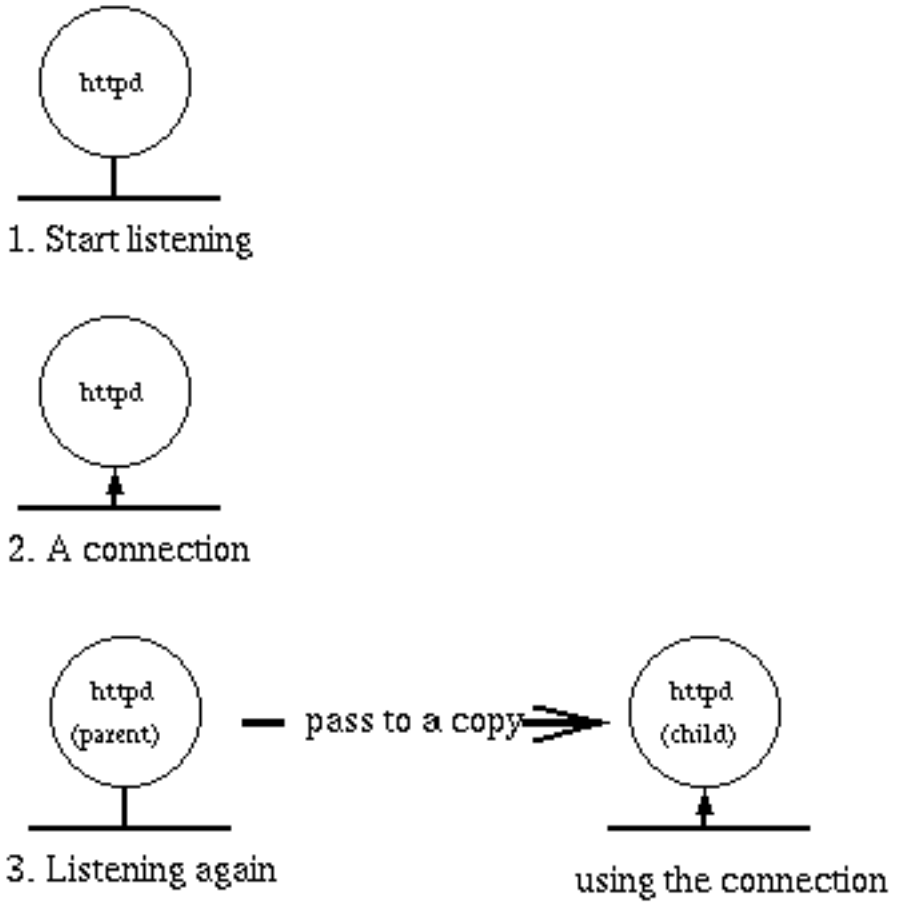
Cascata di processi e processi figli in Apache

Un tipico albero dei processi di ***Apache*** è qualcosa di simile a quello illustrato nella tabella seguente: lo *user* è il proprietario del **processo**, ed il primo *threed* (*root*) è il **processo** lanciato per avviare il *Web* ***server***.

USER	*PID*	*%CPU*	*%MEM*	*SIZE*	*RSS*	*TTY*	*STAT*	*START*	*TIME*	*COMMAN*
root	203	0.01	1.01	4952	720	?	S	17.20	0.03	/usr/sbin/apache
user	212	0.00	2.03	5012	1456	?	S	17.20	0.06	/usr/sbin/apache
User	213	0.00	2.02	5008	1424	?	S	17.20	0.06	/usr/sbin/apache
user	214	0.00	0.00	4976	0	?	SW	17.20	0.00	_(apache)
user	216	0.00	0.00	4976	0	?	SW	17.20	0.00	_(apache)
user	473	0.00	1.06	4976	1072	?	S	18.05	0.00	/usr/sbin/apache
user	477	0.00	1.06	4976	1076	?	S	18.05	0.00	/usr/sbin/apache
user	478	0.00	2.04	5012	1544	?	S	18.05	0.00	/usr/sbin/apache

Una volta avviato il **processo** *root*, questo legge la sua configurazione e quindi si mette in ascolto di connessioni ***client***, tipicamente sulla porta 80. Il ciclo di funzionamento di *apache*, per altro identico a quello di ogni altro *Web* ***server***, può essere riassunto nei punti seguenti:

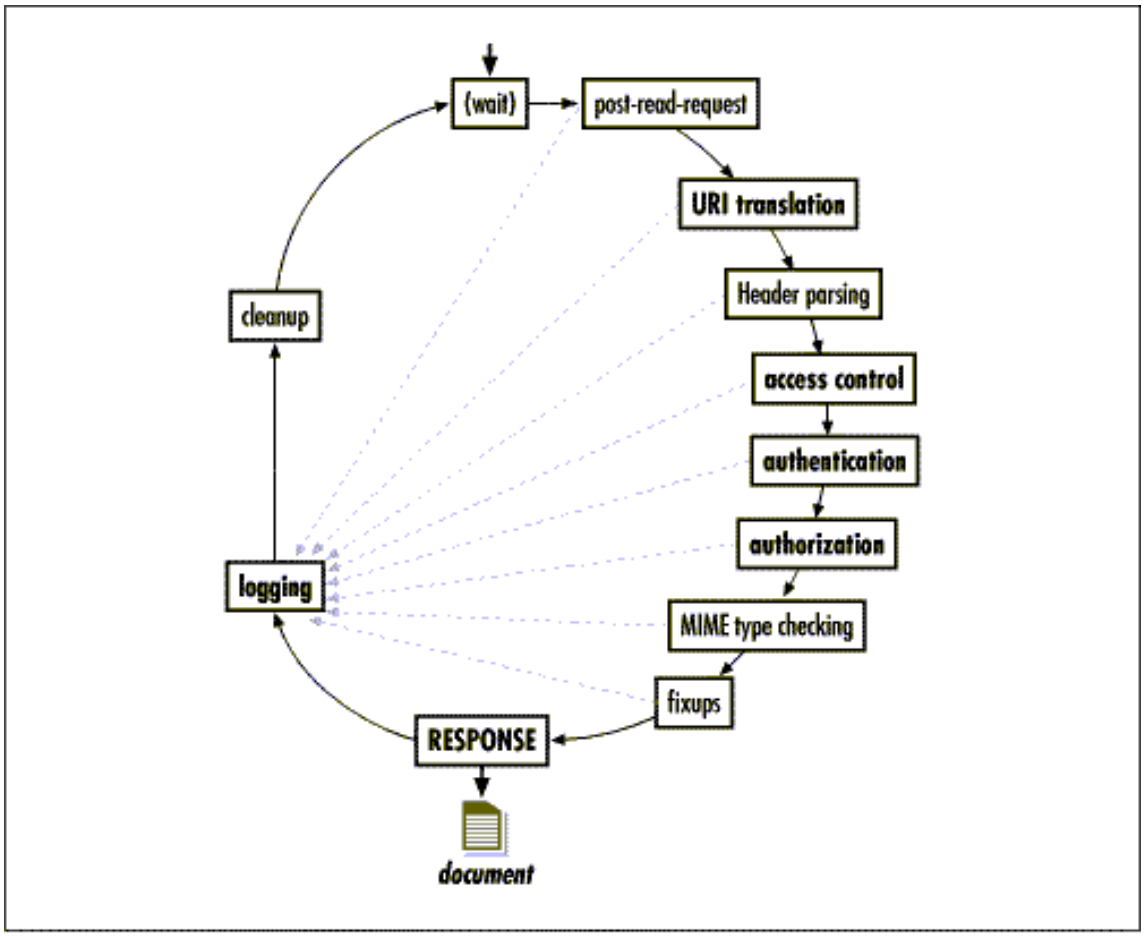
- Una volta avviato, il ***server*** è un demone in stato di attesa (*listening*) di una connessione ***client***.
- La richiesta di un ***client*** apre una connessione col ***server***, il quale è in grado di soddisfare soltanto quella. Per soddisfare le successive il demone lancia un **processo** figlio a cui fa gestire la ***socket*** *TCP*.
- Il demone è quindi ancora attivo e disponibile a stabilire nuove connessioni.



Fasi di un processo server Apache

Ciclo di vita del processo

Il vantaggio di una struttura a processi è notevole: il ***server*** è snello (il demone HTTPd occupa poche decine di KB) poiché deve essere in grado di risolvere una sola richiesta alla volta. L'interruzione (o il blocco improvviso) di qualunque **processo** figlio non manda in *tilt* il ***server***, poiché il padre è in grado di continuare a funzionare in maniera indipendente da ogni suo **processo** figlio. Il ciclo di richiesta/risposta che realizza il *Web* ***server*** è illustrato nella figura seguente.

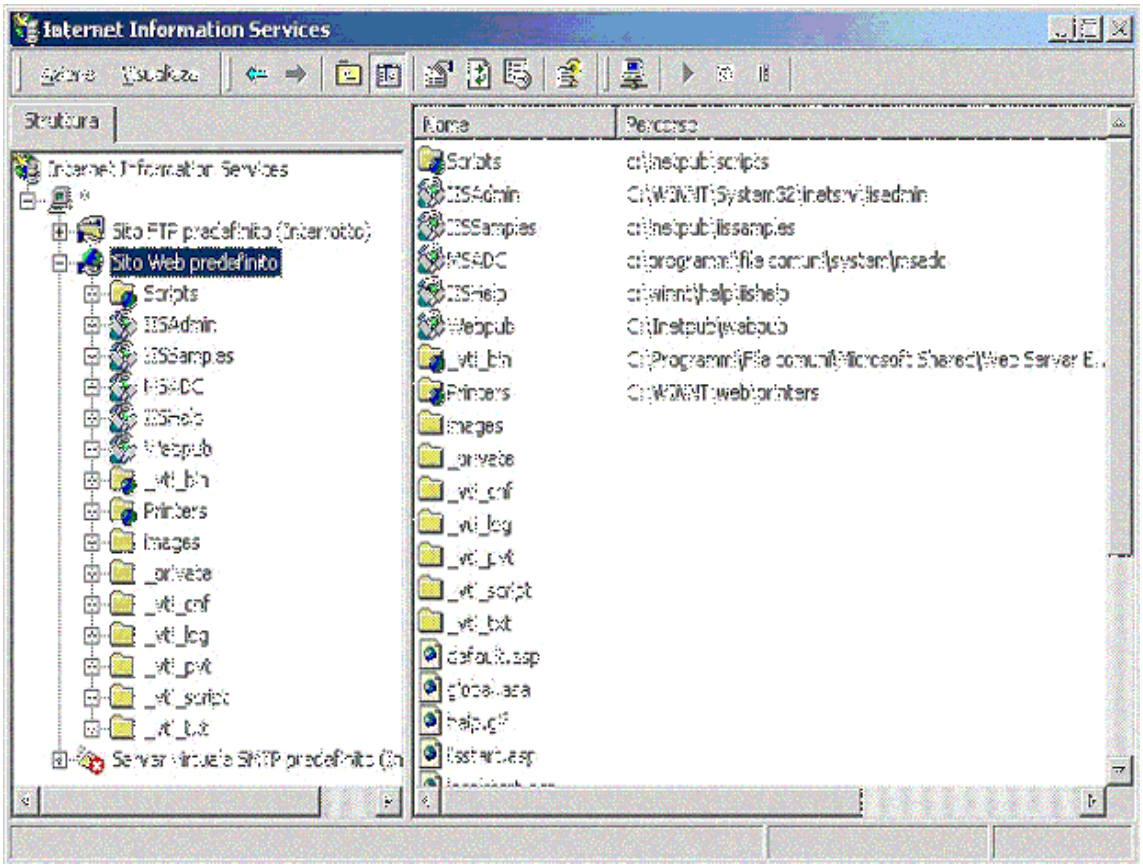


Ciclo di richiesta/risposta di un processo Apache

Ricordiamo inoltre che ***Apache*** è distribuito come *free software*, per esplicito desiderio del *team* che lo sviluppa: questo, infatti, ritiene che strumenti di questo genere debbano essere accessibili a tutti, e che le *software house* debbano guadagnarci solo producendo *addons* o gestendo servizi, magari personalizzandoli per alcune categorie di utenti. Inoltre, lasciando il *software* libero e completo di sorgenti, è più facile controllarne gli errori di sviluppo tramite *feed back*, data la semplicità con la quale si accede al sistema.

Server Web: Microsoft Internet Information Server

Windows 2000 ***server*** comprende un *Web Server* che va sotto la denominazione di *Internet Information Services* (**IIS**). **IIS** viene installato di *default* come servizio di rete durante l'installazione di *Windows* 2000 e permette di supportare anche configurazioni di *Web Server* abbastanza complesse (*Server Web* Virtuali, Cartelle Virtuali, ...). *Internet Information Server* (**IIS**) di *Microsoft* consente di creare applicazioni che fanno uso di pagine *Web* dinamiche per utilizzo pubblico su ***Internet*** o all'interno di reti aziendali. È una piattaforma per applicazioni ***Internet***/*Intranet*, utilizzabile soltanto in ambiente *Windows* NT, o superiori. Le applicazioni sviluppate sono compatibili con qualsiasi *browser Web*, funzionante con qualsiasi sistema operativo su ***client***.



Una finestra di Microsoft Internet Information Server

IIS include un efficiente motore **HTTP**, servizi **FTP**, SMTP e *Gopher*. Esso è costituito da una serie di pacchetti applicativi, che offrono un ambiente per sviluppare applicazioni, un motore di ricerca, un supporto per trattare elementi multimediali. I pacchetti più importanti sono:

- ***Active Server Page*:** permette di creare rapidamente potenti applicazioni *Web* attraverso l'integrazione di elementi **HTML**, ***script*** e *component*. È possibile inserire moduli *Visual Basic* e *JScript*, mantenendo la compatibilità con qualsiasi motore di *scripting*, come *Perl*, e gli altri linguaggi ***CGI***. *Active Server Pages* incorpora inoltre una funzione avanzata di accesso via *Web* a *database* aziendali.
- ***Microsoft NetShow*:** è una piattaforma *software* aperta per la trasmissione, sia in tempo reale che su richiesta, di contenuti multimediali su ***Internet*** e sulle *Intranet* aziendali. Per le trasmissioni in tempo reale, usa la diffusione selettiva (*multicasting*) per distribuire simultaneamente *file* audio e dati; per quanto riguarda invece la trasmissione su richiesta, permette di memorizzare e inviare contenuti audio, video e *Illustrated Audio* (audio sincronizzato con immagini, indirizzi **URL** e ***script***). Inoltre, *NetShow* supporta l'*ActiveMovie Streaming Format*, che assicura avanzate funzionalità multimediali per *authoring* e sincronizzazione.
- ***Microsoft Index Server*:** è un motore di ricerca integrato, in grado di indicizzare automaticamente testi completi e proprietà dei vari *file*,

compresi quelli in formato **HTML**. *Index Server* riesce inoltre a reperire tutti i documenti indipendentemente dal tipo di formato, che può essere un documento *Microsoft Word*, un foglio elettronico *Microsoft Excel*, o una pagina **HTML**. L'indice viene dinamicamente aggiornato quando i documenti cambiano, mentre le funzionalità di sicurezza sono strettamente integrate con quelle di *Microsoft Windows NT*.

- ***FrontPage Server*:** permette la creazione e la gestione delle pagine di un sito ***Internet***/*Intranet*. È uno strumento *client/server* visuale facile e potente per sviluppare e mantenere aggiornati siti *Web*. L'interfaccia utente è coerente con quella di *Microsoft Office* pertanto è facile da usare. La sua architettura *client/server* consente la creazione di pagine *Web*, la preparazione di ***script*** di comandi e la gestione di un sito *Web* anche da una macchina remota.

I benchmark sui server Web

Diversi produttori e riviste si sono avventurati in confronti sui prodotti di gestione dei ***server*** *Web*, creando non poca confusione. Questi confronti, detti ***benchmark***, sono test che intendono mostrare le *performance* di un prodotto con *software* e *hardware*.

I due *Web* ***server*** maggiormente diffusi sulla rete sono, senza dubbio alcuno, *Microsoft Internet Information Server* (**IIS**) e ***Apache***. La differenza principale fra i due applicativi è la piattaforma per la quale sono stati pensati: **IIS** è per ***server*** basati su sistemi *Windows*, ***Apache*** per ***server*** basati su *Unix*.

Di seguito sono discussi alcuni ***benchmark*** effettuati utilizzando macchine simili con configurazioni *Linux*/***Apache*** e *Windows* NT/**IIS**.

Il primo test di confronto tra ***Apache*** e *Microsoft* **IIS** è stato effettuato facendo richiedere a tutti i ***client*** la stessa pagina **HTML** da 4Kb: il risultato è una sostanziale parità di *performance* fra i due *Web* ***server***.

Il secondo, invece, si basa su una richiesta casuale da parte dei ***client*** di una pagina fra le 10.000 (ovviamente della stessa grandezza) presenti in una *directory* del ***server***. In questo test, *Linux*/***Apache*** è risultato circa il 15% più veloce.

Il terzo test, che inizia ad essere impegnativo per le macchine, si basa sulla richiesta di una quantità di *file* di due volte superiore a quella della RAM dei ***server*** (quindi qualcosa come 4 GB di pagine richieste). NT/**IIS** sembra non riuscire a sopportare più di 30 richieste al secondo, mentre *Linux*/***Apache*** arriva a 166. La differenza, si commenta, può essere principalmente dovuta al tipo di partizione utilizzata dai sistemi (in termine di grandezza dei *cluster* o *i-node*) che non alle capacità degli stessi.

Il quarto test si basa non più su pagine statiche come nei precedenti tre test, ma su pagine dinamiche: la scelta è ricaduta sui ***CGI***, non essendo le tecnologie **ASP** e *VBscript* direttamente portabili ad altri sistemi. Su NT è stato installato *ApcivePerl* 517.3, equivalente al *Perl* 5.005_3 presente sul ***server*** *Linux*.

Il risultato è palese: NT/**IIS** soffrono della non-natività del linguaggio *Perl* utilizzato per l'interpretazione dei ***CGI***, riscontrabile come un'eccessiva lentezza nell'invio delle pagine create dinamicamente. Ma per questo test, avvertono, il risultato era scontato dall'inizio, e quindi poco significativo. Ricordiamo infatti che il *Perl*, linguaggio più

largamente utilizzato per scrivere *script* ***CGI***, è nato in ambiente *Unix* e, sebbene il *porting* di *Activestate* per le piattaforme *Windows* sembra essere molto ben riuscito, ancora una volta la non-natività del linguaggio di interpretazione si fa sentire.

L'ultimo test è quello di servire sedici macchine tramite due schede di rete. Qui NT/**IIS** riesce a prevalere sull'avversario *Linux*/***Apache*** anche con una potenza di calcolo minore, ossia con un processore in meno. La velocità con cui NT/**IIS** invia le pagine ai ***client*** ha fatto addirittura pensare che le *performance* non sarebbero peggiorate significativamente neanche con quattro schede di rete al posto di due.

Le conclusioni degli autori del test sono le seguenti: per una rete mista, formata cioé da richieste differenti ed indipendentemente dal volume delle stesse, l'accoppiata *Linux*/***Apache*** risulta migliore. Per un ***server*** con più schede di rete e con prestazioni medio-alte, NT/**IIS** è l'accoppiata ideale. Ovviamente, le necessità in pratica possono essere anche altre, ad esempio il **linguaggio di *scripting*** da adottare (***CGI***, **ASP**, *VBscript*, eccetera), la potenza della macchina da utilizzare (sembra infatti che *Linux*/***Apache*** riesca a spremere maggiormente le *CPU*, sebbene NT/**IIS** appaia sempre più performante per ogni *CPU* aggiunta), eccetera.

Installazione e configurazione di un Web server

Il **processo** di creazione di un *Web Server* è un **processo** molto simile a quello relativo alla creazione di un *file* ***server***. Come nel caso di ogni altra tipologia di ***server*** bisogna innanzitutto installare il sistema operativo, decidere l'appartenenza ad un gruppo di lavoro (*stand-alone* ***server***) o ad un dominio (*member* ***server***), ed organizzare in maniera opportuna i dischi. Il passo successivo consiste nell'installare e configurare in maniera appropriata i servizi relativi alla funzionalità di *Web Server*. Ricordiamo che tramite un *Web Server* un ***client*** accede a *file* che costituiscono le pagine *Web*, ma anche a complesse applicazioni *client/server* basate su *database*.

Brevemente, per configurare un *Web* ***server*** sono necessari:

- Installare un *TCP/IP*.
- Un **indirizzo** *IP* statico.
- Un nome di domino.

Installazione e configurazione di Apache

Ci sono due possibilità di installazione per ***Apache***: il nuovo **processo**, detto APACI, è più veloce e comodo, poiché utilizza tutte le modalità standard di installazione dei pacchetti sotto *Linux*; il **processo** più vecchio è invece quello manuale, lungo e noioso. Vediamo come si svolge il tutto per il primo:

- dobbiamo innanzitutto assicurarci di avere risorse disponibili: dobbiamo avere circa 12 Mb di spazio temporaneo su disco e 3 Mb necessari all'installazione; serve poi il compilatore, che presumibilmente avrete già installato: è vivamente consigliato GCC 2.7.2 o superiore. Vengono poi citati come opzionali l'interprete *perl* 5 (che comunque dovrete avere, altrimenti i ***CGI*** saranno impossibili da eseguire) ed il supporto per i DSO (*Dynamic Shared Object*, che servono per alcune chiamate di sistema per il caricamento dei moduli esterni);

- lanciamo lo ***script*** ./configure, per preparare la compilazione dei sorgenti; tale ***script*** accetta varie opzioni:
  - --prefix=PREFIX. È la *directory* nella quale volete installare ***Apache*** (ovviamente *PREFIX* dovrà essere cambiato con il nome della *directory*): consigliata /usr/local
  - --add-module=FILE. Serve per copiare il sorgente di un modulo nell'albero dei sorgenti di ***Apache***; ovviamente al posto di *FILE* dovrete inserire il *path* per il sorgente del modulo.
  - --activate-module=FILE. Aggiunge al volo un'*entry* per un modulo al *file* di configurazione di ***Apache***.
  - --enable-module=NAME . Serve per fare in modo di abilitare (o disabilitare, utilizzando *--disable-module*) un modulo particolare.
  - --with-perl=FILE . Serve per impostare l'interprete perl utilizzato da ***Apache*** (anche se di solito ***Apache*** cerca da solo l'interprete).
- Ovviamente la opzioni da passare a ./configure non sono tutte qui: si è pensato di presentarvi solamente le più utili (soprattutto al nostro scopo). Per conoscere tutte le opzioni, lanciare './configure --help'.
- Iniziamo la compilazione lanciando 'make'.
- In seguito si installa: *make install* è il comando da lanciare.
- A questo punto ***Apache*** è installato ma non ancora attivo: lanciamo PREFIX/sbin/apachectl start e dovremmo poter richiedere ad ***Apache*** il primo **URL**, che sarà http://localhost. Se volete fermare ***Apache***, lanciare PREFIX/sbin/apachectl stop. Per la configurazione il *file* principale è /etc/apache/HTTPd.conf, ma è possibile averlo da qualche altra parte, soprattutto se si è partiti compilando i sorgenti.

Configurazione di Apache

Vediamo com'é strutturato *HTTPd.conf*: intanto, ogni voce è ampiamente commentata, così da farvi capire cosa state facendo e a cosa serve quello che state facendo. Le voci più significative sono:

Voce	Descrizione
ServerType	Il tipo di ***server***, '*standalone*', è la impostazione *standard*
Port	È la porta. Di *default* ha valore 80
HostnameLookups	Logga i nomi dei ***client*** (*on*) o solamente il loro numero *IP* (*off*); lasciatelo pure su *off*
ServerAdmin	Se si imposta un indirizzo *e-mail*, su ogni errore, verrà incluso nei messaggi inviati ai ***client***.
ServerRoot	La *directory* dove ***Apache*** conserva i *log*, gli errori e i *file* di configurazione; solitamente è la *directory* dove è presente anche il *file* HTTPd.conf
LoadModule	Indica ad ***Apache*** quali moduli caricare. Indispensabili sono tutti i moduli per i ***CGI***, per il *Perl* e i *mime* (commentati)
ErrorLog	Il *file* dove ***Apache*** scrive gli errori. /var/log/apache/error.log è quello di *default*
ServerName	Il nome del vostro ***server***. Se non impostato, sarà localhost.localdomain ma, siccome è assai scomodo, lanciate (da *root*) *hostname* e cambiate il nome al vostro ***host***

Un altro importante *file* da configurare è srm.conf, che dovrebbe essere nella stessa *directory* di HTTPd.conf.

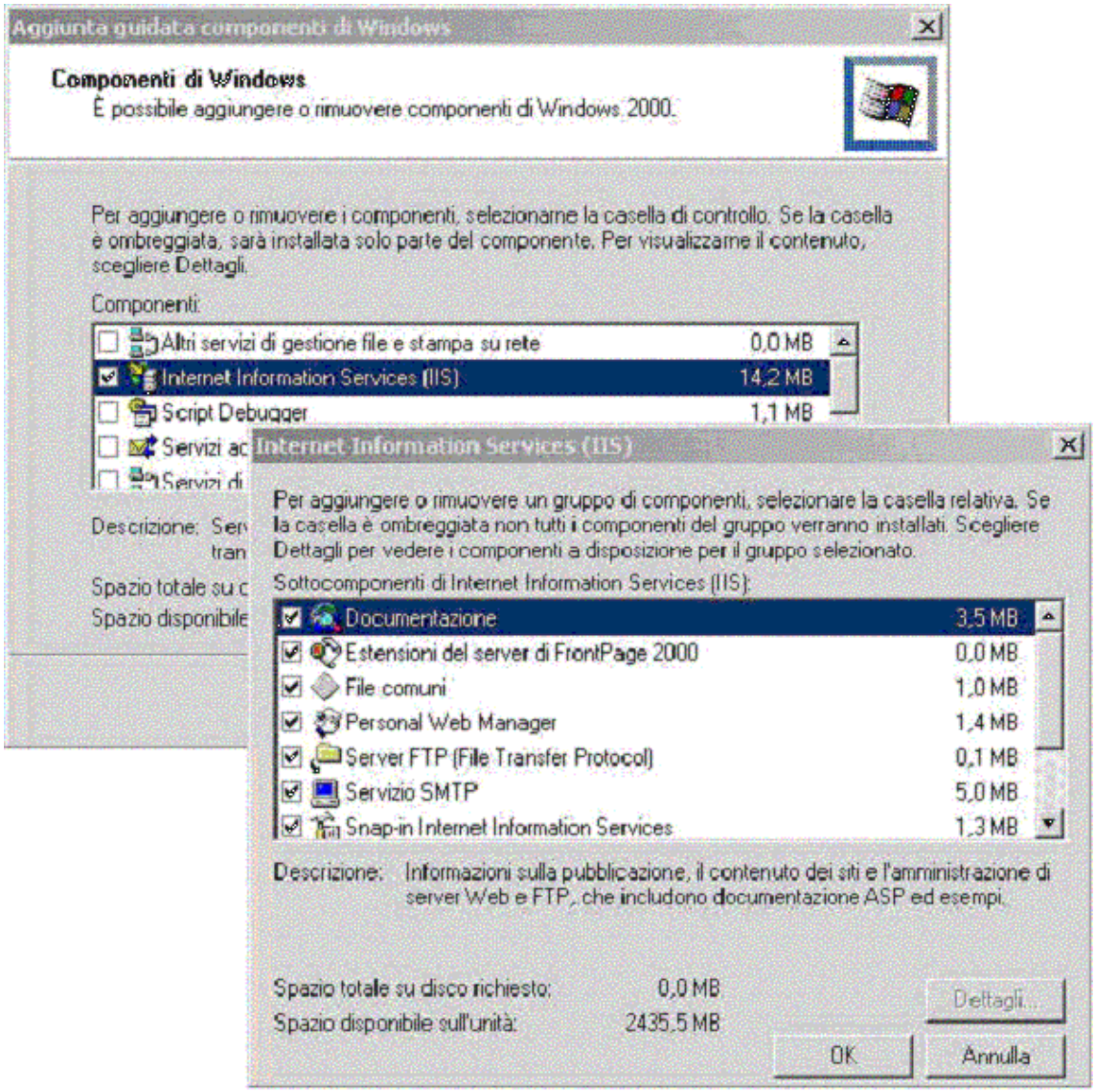
Voce	Descrizione
DocumentRoot	La *directory* nella quale mettere i *file* html per la pagina locale. Solitamente /var/www: quindi se in tale *directory* mettete un vostro *index.html*, indicando '*localhost*' come **indirizzo** del *browser*, vedrete proprio questa pagina. E da questa partiranno le altre pagine
DirectoryIndex	Il nome della pagina che verrà visualizzata come indice; solitamente *index.html*
ScriptAlias	Ed eccoci arrivati ai ***CGI***: conviene impostare tale voce come: ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/. Cosa significa? Semplicemente, che voi potrete inserire i vostri ***CGI*** in /usr/lib/cgi-bin/, ma questi saranno chiamati tramite http://localhost/cgi-bin/nome_cgi.cgi

Se qualcosa non dovesse funzionare al meglio, leggete la documentazione inclusa

nell'archivio che avete scaricato ed eventualmente cercate la cartella /usr/doc/apache.

Installazione e configurazione di Web server: Microsoft IIS

Fare clic sul pulsante *Start*, scegliere Impostazioni, quindi Pannello di controllo. Fare doppio clic sull'icona Installazione applicazioni, scegliere Configurazione di *Windows*, quindi Componenti. Seguire le istruzioni visualizzate per installare, rimuovere o aggiungere componenti di **IIS**.



Finestre di Windows per guidare l'aggiunta della componente Microsoft IIS

Come installare *Visual Interdev*

Se si desidera disporre di un ***server*** locale, installare **IIS** servendosi dello strumento Installazione applicazioni prima di installare *Visual InterDev*. Se si installa *Visual*

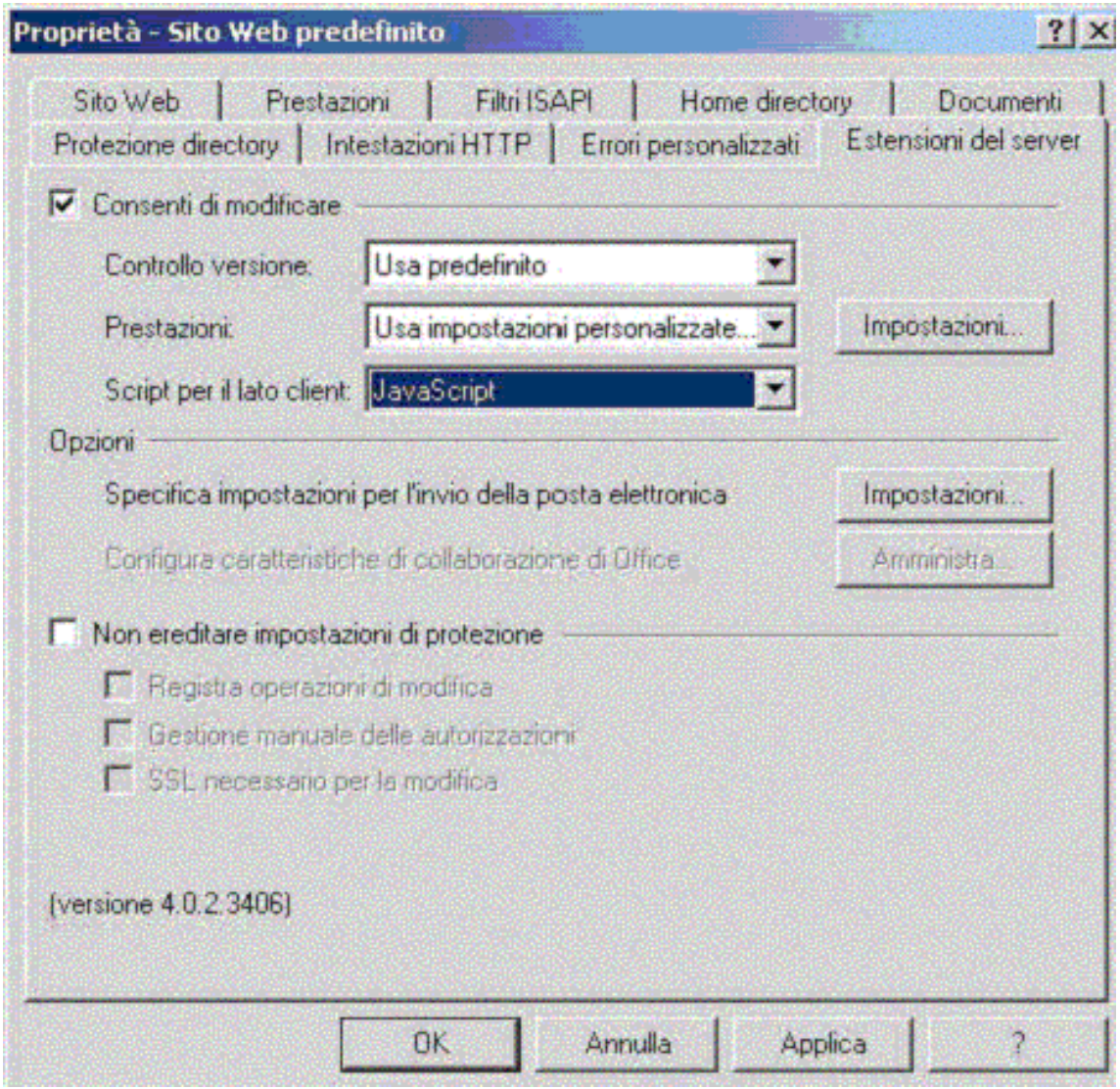
InterDev 6.0 in un *computer* in cui è installato *Visual InterDev* 1.0, è necessario disinstallare la versione 1.0 prima di installare la versione 6.0. Se sono già stati installati componenti di *Visual* Studio prima di installare *Visual InterDev*, è necessario avviare l'installazione di *Visual InterDev* facendo clic su *Microsoft Visual* Studio versione del prodotto nello strumento Installazione applicazioni del Pannello di controllo e scegliendo Aggiungi/Rimuovi.

Dopo avere installato il ***client*** di *Visual InterDev* sul *computer*, viene richiesto di installare *Microsoft Developer Network* (MSDN). È necessario installare MSDN perché il tasto F1 della Guida funzioni e si possa accedere alla documentazione di *Visual InterDev*. Se MSDN è già stato installato e non si desidera installarlo, deselezionare la casella di controllo Installa MSDN, quindi scegliere Fine.

Installare le estensioni del server

Per installare le estensioni del ***server*** di *FrontPage* 2000 per *Microsoft Internet Information Server*, fare doppio clic su *file* eseguibile scaricato dal sito *Web Microsoft* e seguire le richieste visualizzate. Le estensioni del ***server*** di *FrontPage* verranno configurate esclusivamente per il sito *Web* predefinito. È possibile effettuare l'istallazione soltanto utilizzando l'*account* di amministratore o un *account* del gruppo *Administrators*. È possibile scaricare le estensioni del ***server*** di *FrontPage* 2000 dal sito *Web Microsoft*: http://msdn.microsoft.com/workshop/languages/fp/2000/winfpse.asp.

Scaricare ed installare il *file Fpse2k_x86_eng.exe* (circa 14 MB). È necessario effettuare un riavvio del *computer*, in seguito al quale verranno eseguite altre operazioni di installazione. Fare doppio clic su Strumenti di amministrazione, quindi fare doppio clic su Gestione servizio *Internet Microsoft*. Fare clic sul segno più + accanto al nome del ***server*** per espandere il ramo. Fare clic con il pulsante destro del *mouse* su Sito *Web* predefinito, scegliere Tutte le attività, quindi configurare le estensioni del ***server***. Nella Configurazione guidata estensioni del ***server***, scegliere Avanti.



Finestra delle proprietà del sito Web con la selezione della cartella per le estensioni del server di FrontPage 2000 per Microsoft Internet Information Server

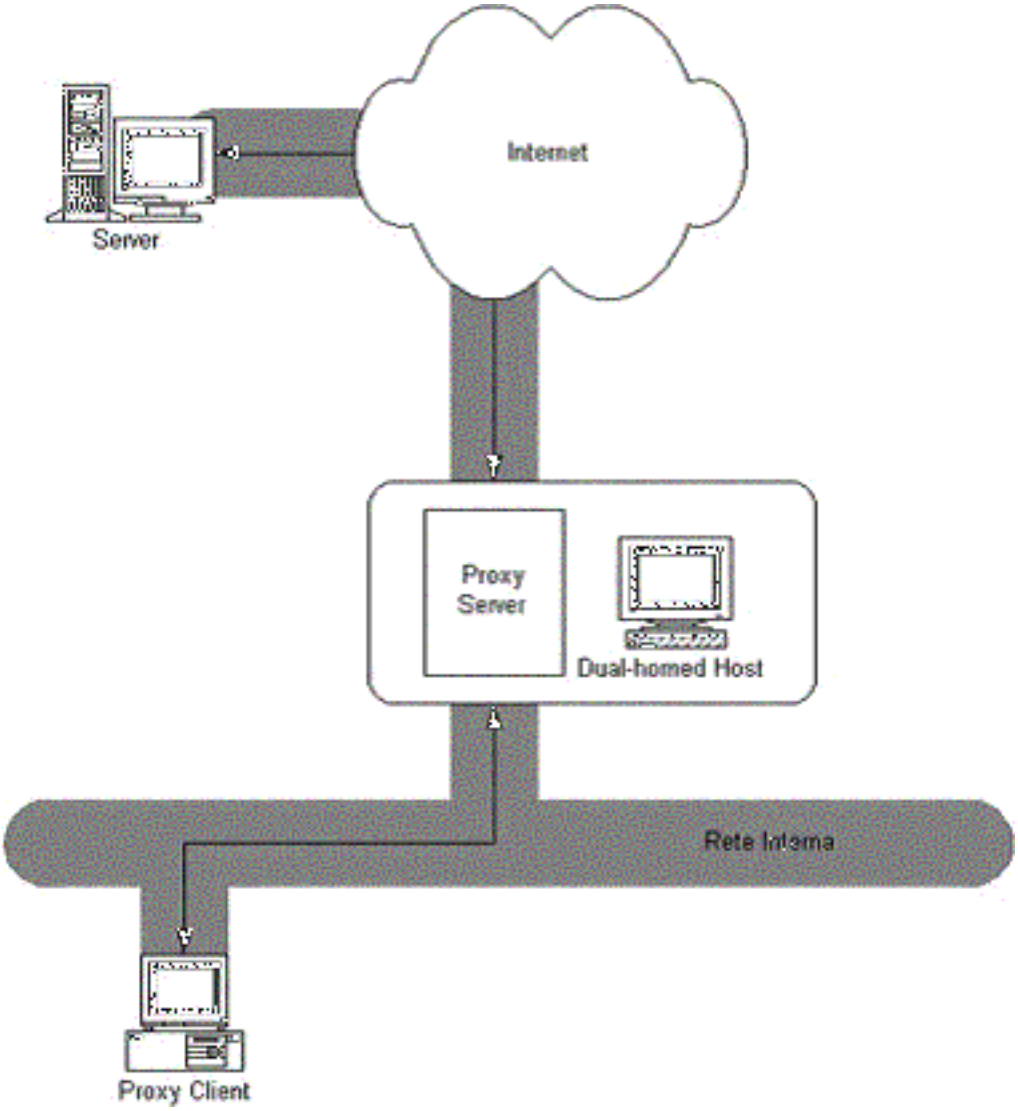
Attenzione: scegliere No se viene visualizzato il seguente messaggio: Se si mantiene la configurazione corrente del ***server*** *Web*, la modifica delle pagine disponibili nel ***server*** dopo l'installazione delle estensioni sarà consentita a chiunque. Continuare con l'installazione delle estensioni?

A questo punto è necessario configurare la partizione in cui è residente **IIS** in modo che utilizzi il *file system* NTFS anziché FAT o FAT32 (da evitare accuratamente soprattutto sulle macchine di produzione). Per risolvere questo problema sospendere l'installazione delle estensioni e riconfigurare **IIS** in modo che utilizzi una partizione NTFS. Configurare le impostazioni del ***server*** di posta digitando il nome del ***server*** SMTP, quindi scegliere Avanti. Si noti che è necessario effettuare questa configurazione solo se dal sito *Web* verranno inviati messaggi di posta elettronica, mentre non è necessaria per la maggior parte delle funzionalità **ASP**.

Descrizione dei proxy server, installazione e

configurazione - I proxy server

Un ***proxy*** *server* è un *computer* che funge da interfaccia nelle connessioni a ***Internet*** tra l'utente e ***Internet*** stessa, fornendo importanti funzionalità di rete.



Relazioni tra server, proxy server e utenti

I ***proxy*** *server* forniscono essenzialmente quattro funzionalità:

- *Firewall* e filtraggio dei messaggi.
- Condivisione della connessione tra più utenti.
- Memorizzazioni delle pagine in memoria *cache*.
- *Gateway* per connettere la rete locale a ***Internet***.

I ***firewall***, come vedremo nel **capitolo successivo** dell'introduzione, consentono il passaggio di messaggi in entrata solamente a repliche di messaggi in uscita precedenti, per evitare di comprometterne la sicurezza. I filtri invece consentono ad un amministratore di disabilitare l'accesso a particolari domini. La condivisione della

connessione tra utenti consente l'utilizzo della stessa connessione ad ***Internet*** da parte di più utenti. Il *caching* consente di migliorare la qualità del servizio perché non spreca la banda di comunicazione della rete, produce risposte più veloci, e permette l'accesso a pagine *Web* anche quando il sito corrispondente è inattivo. Infine, per il servizio di *gateway* della rete locale a ***Internet***, facciamo un esempio. Supponete che il vostro istituto abbia sottoscritto degli abbonamenti a pagamento presso riviste. Naturalmente questi siti sono accessibili dal *computer* sulla vostra scrivania in ufficio. Da casa invece, in cui tipicamente siete nel dominio dell'*Internet Provider*, non c'è possibilità di accedervi. Se non settando opportuni parametri del vostro *browser* per accedere al ***proxy*** *server* dell'istituto (quando questo lo consente!).

Proxy Server e browser

I ***proxy*** *server* operano con specifici protocolli di rete, di cui **HTTP** è il più importante ed il più critico perché consente di accedere a pagine *Web*. Questi protocolli sono:

- **s-HTTP**, o *secure*-HTTP, che consente comunicazioni **HTTP** criptate e sta diventando sempre più comune soprattutto nei siti di commercio elettronico per espletare transazioni economiche (che prevedono il numero di carta di credito, per esempio). Non va confuso con **ssl**, che è un **protocollo** più a basso livello utilizzato da **s-HTTP** ma che non ha niente a che fare con il *server* ***proxy***.
- **FTP** (*File Transfer Protocol*), che consente il *download* o l'*upload* di *file* dalla/alla rete. Il protocollo **FTP** tratta *file* in formato testo o in formato binario ed è tuttora molto utilizzato per scaricare archivi compressi, come i *file* MP3.
- ***socks***, implementa ***firewall*** di sicurezza e viene usato per *chat* e *telnet*.
- ***gopher*** e ***wais***, sono due protocolli poco usati: sono stati due tentativi di standardizzazione prima di **HTTP**.

Tecnicamente si potrebbero utilizzare ***proxy*** *server* differenti per far fronte ai diversi protocolli. Ma questo significherebbe che i clienti che usano i ***proxy*** *server* devono essere a conoscenza di dettagli di comunicazione. Sicché, gli amministratori quasi sempre configurano i ***proxy*** in modo da operare con tutti i protocolli.

Installare e configurare un proxy server

Sono necessarie due informazioni per specificare manualmente un ***proxy*** *server* nel *browser*:

- il **nome dell'*host*** (come configurato nel **DNS**), oppure il suo numero *IP*;
- il **numero di porta *TCP/IP*** su cui il ***proxy*** *server* sarè in attesa di richieste.

Di solito la porta è la stessa per tutti i protocolli di prima (**HTTP**, **s-HTTP**, **FTP**, eccetera), ed è differente dal numero di porta assegnata ai vari protocolli (spesso 80 per **HTTP**, 21 per **FTP**, eccetera).

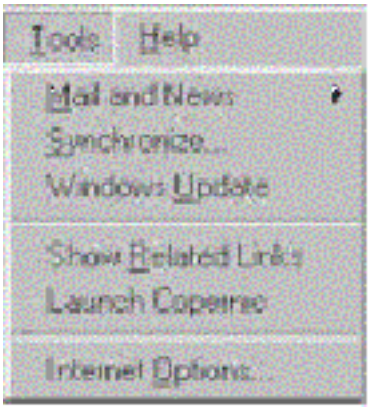
Recentemente, per agevolare gli amministratori nella configurazione dei ***proxy*** *server*, sono stati messi a punto alcuni sistemi. Gli amministratori possono usare opportuni *file* di configurazione con codice *JavaScript* per nascondere agli utenti dettagli quali i numeri di porta. Gli utenti dovranno semplicemente accedere via *browser* a questi *file*

per configurare tutti i parametri.

Microsoft ha messo a punto una nuova tecnologia per *Explorer* (dalla versione 5 in poi), detta *Web Proxy Auto Discovery* (WPAD), per scoprire la presenza di *server* ***proxy*** e di altri servizi *Web*. WPAD usa un servizio di ricerca come **DNS** per creare un *file* di configurazione che gli amministratori possono installare nel *Web* ***server***. Gli amministratori debbono soltanto inserire in questo *file* gli indirizzi dei servizi *Web* relativi.

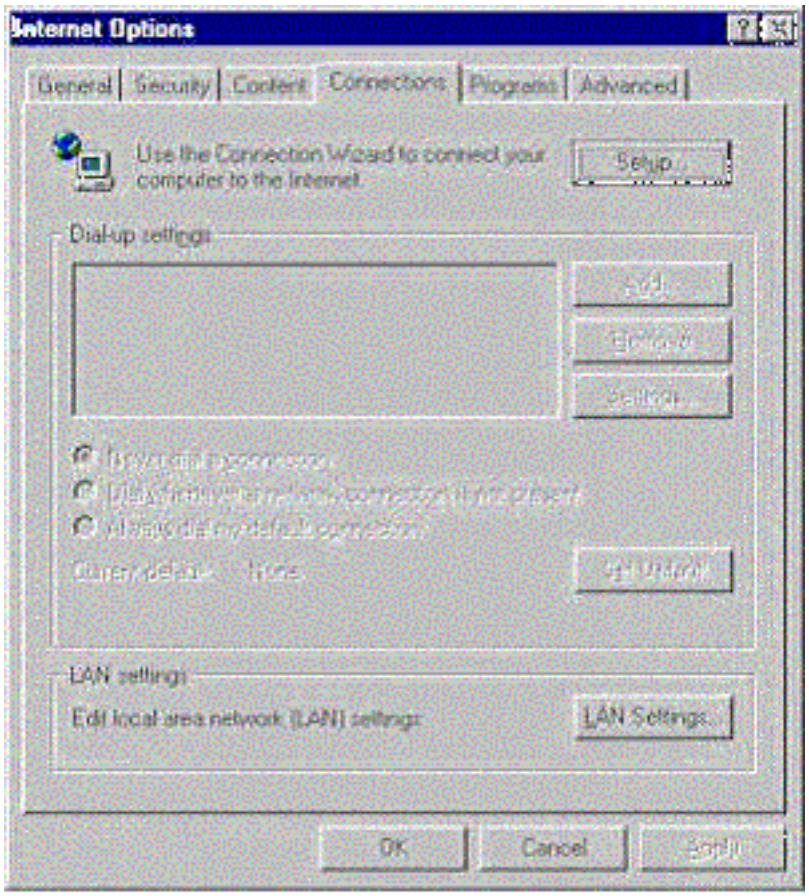
Proxy server e Microsoft Internet Explorer

Per configurare il *Proxy Server* in *Internet Explorer* occorre accedere a *Tool* nel menù di *Internet Explorer*.



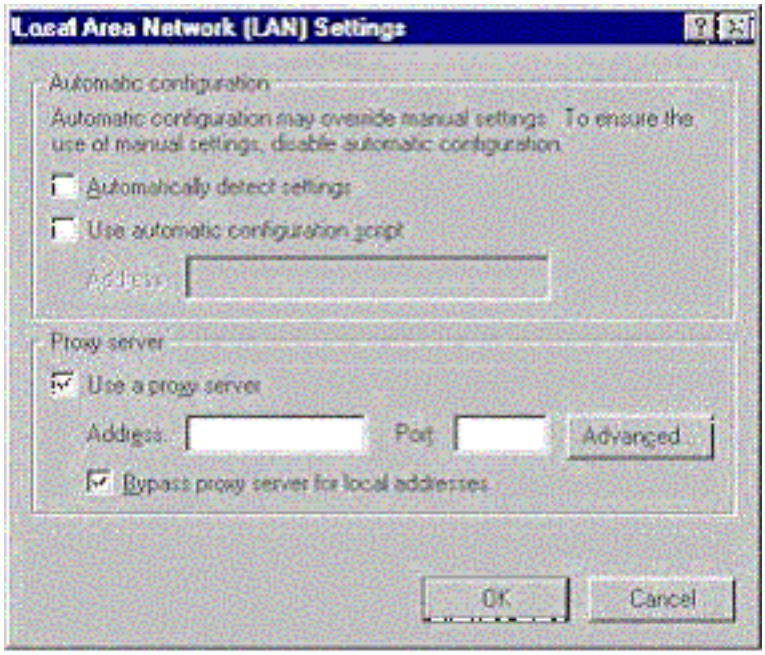
Accesso a Tool nel menù di Internet Explorer

Quindi bisogna selezionare *Internet Options*. A questo punto appare una finestra con diversi fogli, relativi a differenti funzionalitè. Selezionare il foglio delle connessioni di rete (*Connections*).



Finestra di Internet Explorer con la selezione del foglio relativo alle connessioni di rete

A questo punto bisogna selezionare il bottone in basso sui parametri della rete locale (***LAN*** *settings*). Quindi compare una finestra come la seguente:



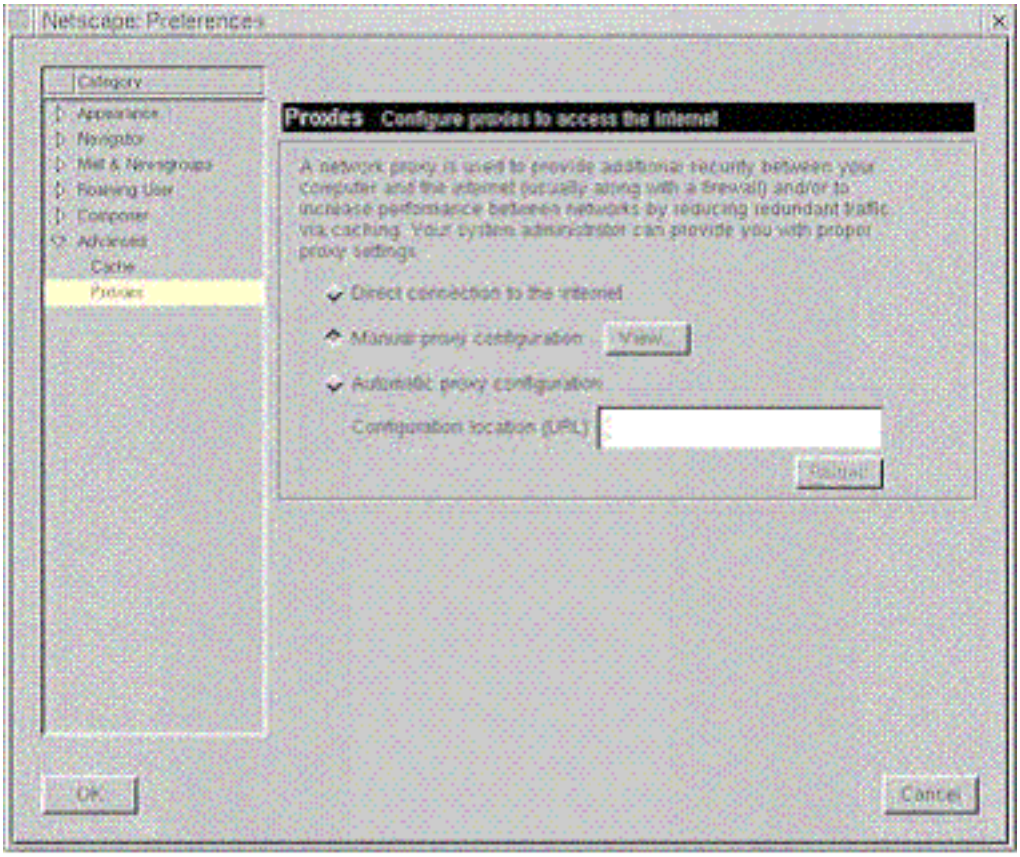
Finestra di configurazione della rete locale e del proxy server di Internet Explorer

In *Internet Explorer* è possibile sia la configurazione automatica che manuale. Per quella manuale bisogna selezionare, come mostrato nella figura di sopra, l'opzione *Use a **proxy** server*. Quindi nel campo di **indirizzo** bisogna inserire il nome dell'***host*** oppure l'**indirizzo** *IP* del ***proxy***. Nella configurazione manuale, bisogna selezionare, nella stessa finestra di sopra, l'opzione *Automatically detect settings* che utilizza il meccanismo WPAD per scoprire la configurazione del ***proxy***. Infine, selezionando *Use authomatic configuration script*, è possibile specificare l'**indirizzo URL** del *file* di configurazione di *JavaScript*.

Proxy server e Netscape Navigator

In *Netscape* si inizia selezionando *Edit* nel menù.

Quindi selezionare *Preferences*. A questo punto compare la finestra di configurazione di *Netscape* che è visualizzata sotto. In questa finestra bisogna selezionare *Advanced* e poi *Proxies*.



Finestra di Netscape con la selezione del foglio relativo ai proxy

Anche *Netscape* consente la duplice configurazione automatica e manuale di ***proxy** server*. Infatti, a questo punto, si può scegliere l'ultima opzione per la configurazione automatica, quella centrale per quella manuale (in tal caso bisogna selezionare *View* per una ulteriore finestra che richiederà il nome del ***proxy** server* o l'**indirizzo** *IP*).

Infine, la prima opzione consente di collegarsi a ***Internet*** in modo diretto, senza utilizzare ***proxy*** *server*.

Firewalls

Cosimo Laneve

Firewall

I ***firewall*** sono una componente o un insieme di componenti che limitano l'accesso tra una rete protetta ed ***Internet***. Essi proteggono le organizzazioni in ***Internet*** fornendo accessi sicuri: garantendo che utenti validi possano accedere alle risorse di rete di cui hanno bisogno.

Determinare chi sia un utente valido è compito del sistema di autenticazione; mentre determinare quali risorse un utente possa accedere è compito del sistema di autorizzazione (*Access Control*). Per fornire meccanismi di *Access Control*, un ***firewall*** richiede una comprensione profonda dei servizi e delle applicazioni utilizzati in rete.

Ci sono fondamentalmente due tipi di ***firewall***, quelli personali e quelli commerciali.

I firewall personali

I ***firewall*** **personali** sono programmi che proteggono un *computer* quando questo è collegato ad una rete. Un *personal* ***firewall*** analizza i canali di comunicazione, negando l'elaborazione del traffico ritenuto rischioso sia in ingresso che in uscita. Di seguito si analizzano le caratteristiche di alcuni prodotti molto diffusi e si riassumono le caratteristiche comparate, in una tabella.

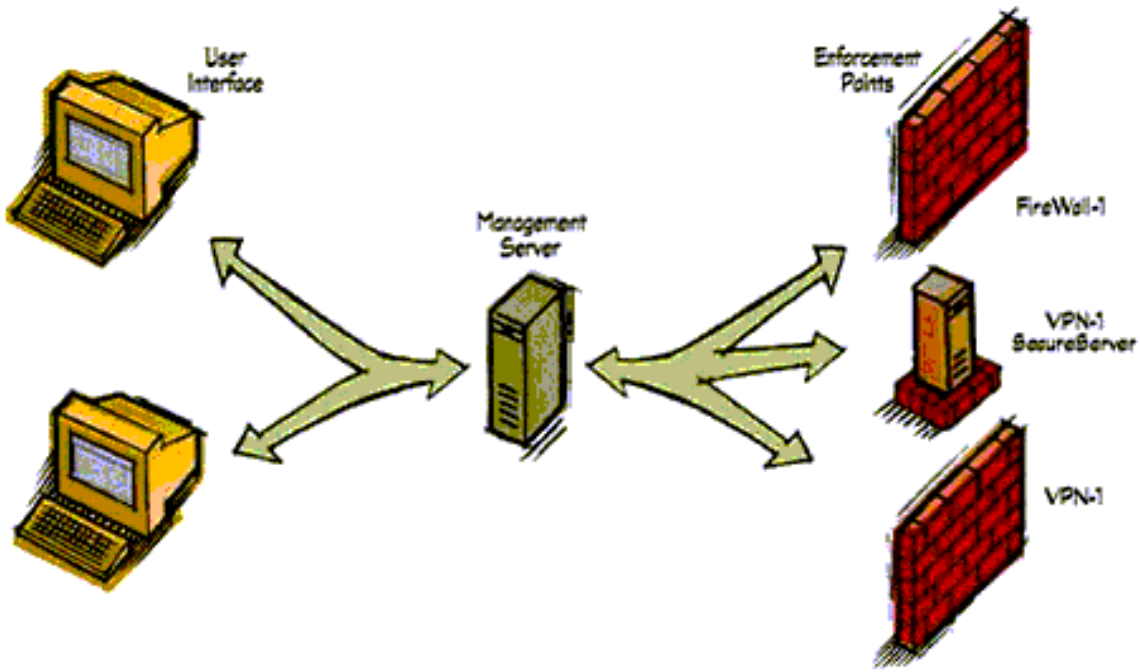
- ***Tiny Personal Firewall*** è un prodotto facile da configurare ed utilizzare che protegge completamente un *computer* dagli attacchi. *Tiny Personal Firewall* include dei *wizard* semplici per il rilevamento delle intrusioni che individuano attività sconosciute e chiedono all'utente di impostare i parametri del ***firewall***. Appositi *wizard* rilevano i tentativi di connessione alle porte di comunicazione e creano delle regole di *filtering* in base all'indicazioni dell'utente. Per garantire che dei cavalli di Troia non si nascondano all'interno di applicazioni viene utilizzata la firma digitale con algoritmo MD5.
- ***Norton Personal Firewall*** è un prodotto che controlla tutte le connessioni tra il *computer* e la rete. Fornisce dei *tool* e dei *wizard* per la configurazione automatica delle regole di *filtering*.
- ***Zone Alarm*** è simile ai precedenti per quel che riguarda protezione e *tool* di configurazione.

La tabella di seguito riportata elenca le caratteristiche dei prodotti considerati, in termini comparativi.

Caratteristica	*Tiny PF*	*Norton PF*	*Zone Alarm*
Firma Digitale	SI	NO	SI
Applicazioni *trusted*	SI	SI	SI
Indirizzi fidati	SI	SI	SI
Rilevamento intrusioni	SI	SI	SI
Amministrazione remota	SI	NO	NO
Log su Syslog	SI	NO	NO
Validità temporale delle regole	SI	NO	NO
Autenticazione	SI	NO	SI
In esecuzione come servizio	SI	SI	SI
Sistemi operativi supportati	95, 98, NT, 2000, Me	95, 98, NT, 2k, Me	95, 98, NT, 2000, Me
Freeware	SI	NO	NO

I firewall commerciali

Una soluzione per la sicurezza di un'organizzazione deve essere in grado di dichiarare una politica a livello di organizzazione, distribuirla e ricevere i *log*. Deve inoltre consentire all'organizzazione di controllare l'intera infrastruttura di sicurezza (i ***firewall*** dell'organizzazione, le reti private virtuali) da un unico punto di amministrazione.

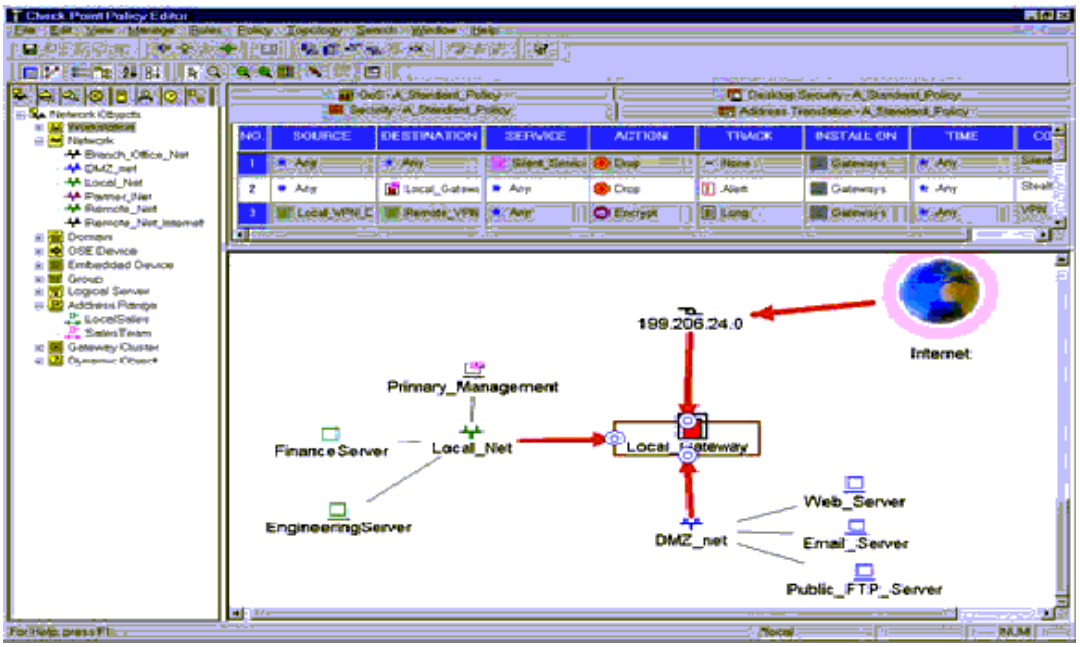


Struttura dei firewall in una organizzazione

Esistono diversi prodotti che soddisfano i requisiti di sicurezza e che forniscono i *tool* per la protezione delle reti private delle organizzazioni. Si analizzano a titolo di esempio le caratteristiche di due prodotti commerciali molto diffusi: *Cisco PIX* e *Checkpoint*

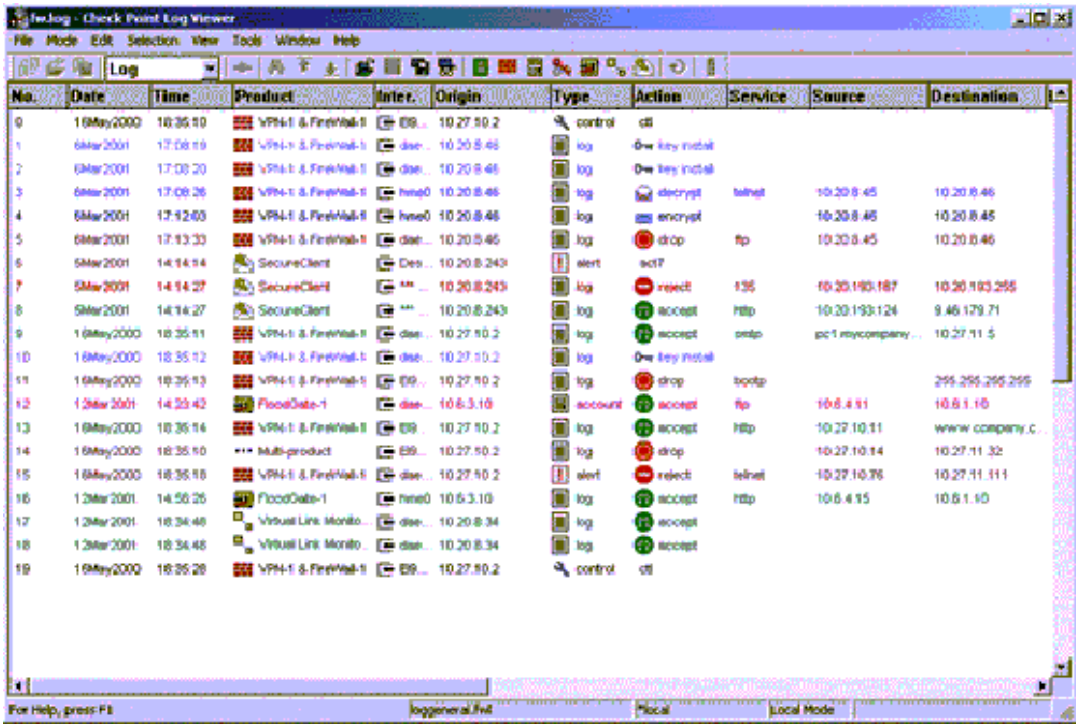
Firewall 1.

- ***Firewall Cisco***. Le principali caratteristiche sono:
  - *Context-Based Access Control*: fornisce agli utenti interni un controllo di accesso sicuro per tutto il traffico attraverso il ***firewall***.
  - Rilevamento delle intrusioni: fornisce il monitoraggio, l'intercettazione e la risposta in tempo reale agli abusi nella rete rilevando un vasto insieme di attacchi comuni.
  - *Proxy* di autenticazione: fornisce meccanismi di autenticazione e autorizzazione degli utenti per quel che riguarda le comunicazioni di rete e/o *dial-up*.
  - Rilevamento e prevenzione di attacchi di tipo DOS: difende e protegge le risorse del *router* da attacchi comuni.
  - Assegnazione dinamica delle porte.
  - Blocco delle ***applet Java***.
  - Supporto per reti VPN, cifratura IPSec e qualità del servizio.
  - *Alert* in tempo reale.
  - Funzionalità di *auditing* dettagliati: memorizza la data, l'***host*** di origine, l'***host*** di destinazione, le porte, la durata e il numero totale di *byte* trasmessi.
  - *Logging* degli eventi: consente agli amministratori di rilevare in tempo reale, potenziali buchi di sicurezza o altre attività non *standard* effettuando il *logging* dei messaggi di errore di sistema su un *Syslog* ***server***.
  - Funzionalità di gestione del ***firewall***: *tool* di configurazione che offre la possibilità di definire passo passo le azioni necessarie per la protezione della rete.
  - Strategie di *filtering* del traffico base ed avanzate.
  - Ridondanza/*fileover*: dirotta automaticamente il traffico ad un *router* di *backup* nell'eventualità in cui il ***firewall*** vada in errore.
  - Funzionalità NAT.
  - Regole per il *filtering* temporizzato.
- ***Checkpoint Firewall-1***. Un ***firewall*** *Cisco* è un dispositivo *hardware* per la protezione di una rete, *Checkpoint Firewall-1* è invece un'**applicazione** *software*. La *console* di *management* di *Checkpoint Firewall-1* fornisce una singola interfaccia grafica per definire e gestire molti elementi di una rete. Tutte le definizioni degli oggetti sono condivise tra tutte le applicazioni.



Console grafica di management di Checkpoint Firewall-1

Gli amministratori della sicurezza possono selezionare la locazione degli oggetti oppure modificarne le caratteristiche utilizzando l'*editor* visuale per la definizione delle politiche di sicurezza. *Checkpoint Firewall-1* fornisce anche un *editor* visuale per i *log* che consente un'analisi in tempo reale delle informazioni relative al *tracking*, al monitoraggio e all'*accounting* di tutte le connessioni. Il modulo per la generazione dei *report* permette agli amministratori di trasformare i dettagliati *log* del ***firewall*** in *report* di gestione che rappresentano le informazioni mediante tabelle e grafici.



Editor visuale dei log di Checkpoint Firewall-1

Installazione e configurazione di firewall

Per essere efficace un ***firewall*** deve essere ben installato e configurato, altrimenti rischia di essere o troppo restrittivo o troppo permissivo. Per evitare installazioni e configurazioni maldestre, prodotti di cui si è detto in precedenza (***firewall*** personali e commerciali) hanno procedure automatiche di installazione: una volta scaricato il *software* da CD o da ***Internet***, è sufficiente lanciare il *set-up* corrispondente.

Windows XP offre un servizio di ***firewall*** per le connessioni ad ***Internet*** (ICF, *Internet Connection Firewall*). Questo *software* permette di limitare le informazioni scambiate tra ***Internet*** e la rete locale, proteggendo anche un singolo *computer*. Descriviamo i passi più importanti per abilitare o disabilitare un ***firewall*** in *Windows XP*. Per effettuare questi passi, è necessario avere l'accesso al *computer* con un *account* di amministratore.

- Selezionare *Start*, scegliere Pannello di controllo e selezionare Connessioni di rete.
- Selezionare il tipo di connessione che si possiede, quindi in Operazioni di rete, selezionare Cambia impostazioni connessione.
- Nella cartella Avanzate in *Firewall* connessione *Internet*, selezionare una delle seguenti opzioni:
  - per abilitare i *firewall*, selezionare la casella di controllo **Proteggi il *computer* e la rete limitando o impedendo l'accesso al *computer* da *Internet***.
  - Per disabilitare i *firewall*, deselezionate la casella di controllo **Proteggi il *computer* e la rete limitando o impedendo l'accesso al *computer***

da *Internet*. Disattivando il *firewall* i *computer* e la rete saranno esposti a intrusioni dall'esterno.

Approfondimento

Aspetti avanzati di Microsoft IIS: Caratteristiche e Amministrazione

Cosimo Laneve

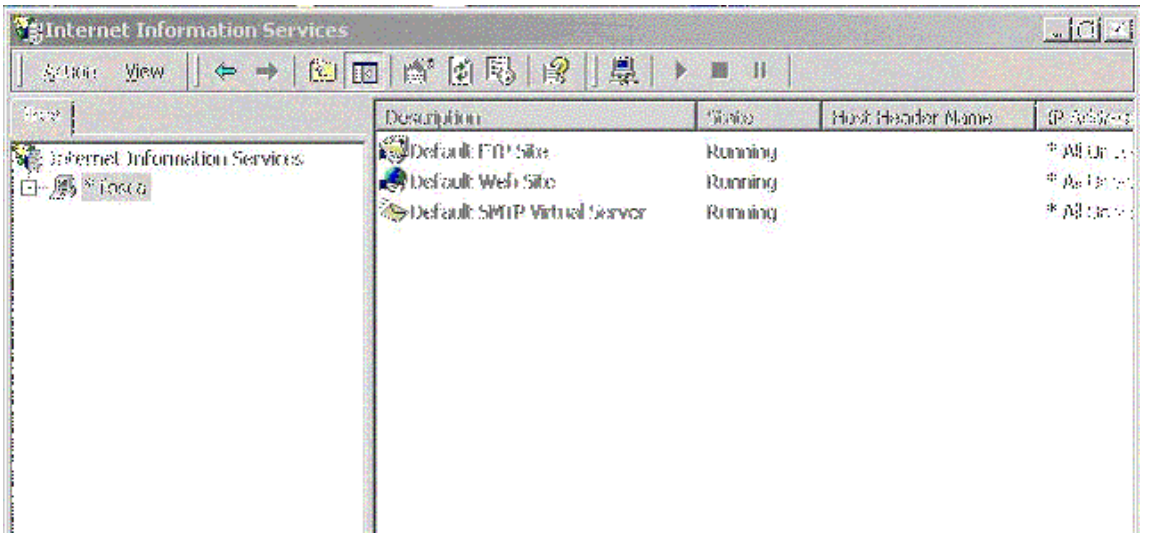
17.1.1 (Installare e configurare un Web server)

Amministrazione di IIS

Il ***server*** *Web* è un **processo** sempre attivo che ascolta richieste **HTTP** su una porta (80 per *default*). Su *Windows* NT un **processo** sempre attivo è chiamato servizio e viene gestito dall'amministratore di sistema dal pannello di controllo. Gestire un servizio vuol dire sostanzialmente attivarlo, fermarlo e configurarne i parametri di funzionamento. Per un *Web* ***server*** tali parametri sono:

- stabilire quali risorse devono essere viste dagli utenti;
- definire quali diritti hanno gli utenti sulle risorse del *Web* ***server***;
- definire il documento di *default* da visualizzare per ogni *directory*;
- stabilire i ***MIME*** *type* (quali applicazioni vanno associate alle estensioni dei *file*).

La finestra di amministrazione di **IIS** si apre dal pannello di controllo in NT e da Pannello di controllo > Strumenti di Amministrazione in *Windows* 2000. La *console* di amministrazione di **IIS** permette di gestire più ***server*** anche in remoto, per *default* comunque viene mostrato il ***server*** locale indicato dal nome del *computer*.



Finestra di amministrazione di Microsoft IIS

IIS mostra una *console* dalla quale si accede non solo al servizio **HTTP** ma anche **FTP** e SMTP. Cliccando su *Default Web site* troviamo tutte le cartelle che sono pubbliche nel nostro sito. Non tutto il disco della macchina su cui gira il ***server*** è visibile all'esterno, ma solo le parti che sono esplicitamente rese pubbliche da chi amministra il ***server*** *Web*. Per *default* tutte le cartelle e i *file* che stanno sotto la *directory Inetpub*

sono pubbliche. L'amministratore può aggiungere altre cartelle alla lista delle *directory* visibili. Per fermare il servizio **HTTP** occorre selezionare il pulsante di *Stop*, per avviarlo sul pulsante di *Start*. Questa operazione però serve solo a fermare il funzionamento, ma il servizio rimane attivo e caricato in memoria.

Con il pulsate *Advanced*, si accede alla configurazione di siti multipla (non solo il predefinito!). Se non si specifica l'*IP address*, tutte le *directory* virtuali saranno visibili a tutti i ***server*** virtuali. In caso si installino più siti sullo stesso *Web* ***server***, è necessario utilizzare per i nomi una tra due alternative:

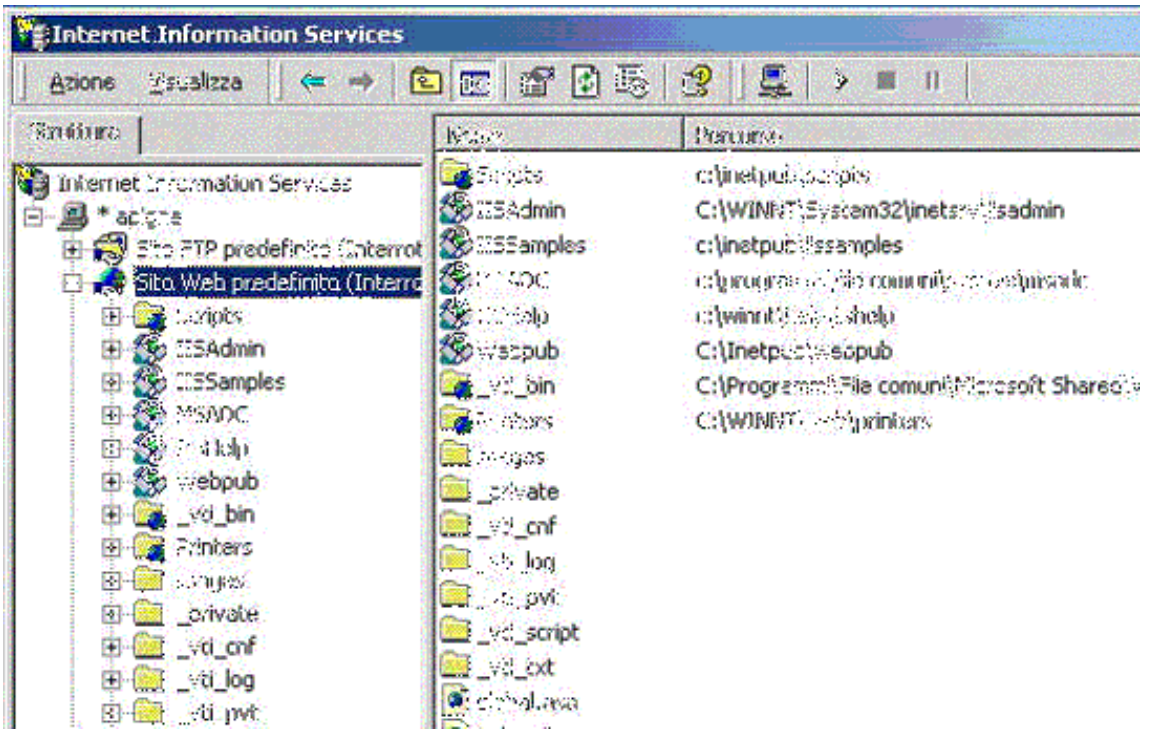
- un **indirizzo** *IP* sulla scheda di rete per ogni ***server*** virtuale.
- Una porta per ogni ***server*** su uno stesso **indirizzo** *IP*.

La prima tecnica è la più utilizzata e non richiede necessariamente *IP* pubblici.

Servizio WWW

Selezionato il sito *Web* sul quale siamo interessati a lavorare notiamo che al suo interno possono essere presenti tre tipi di oggetti:

- **cartelle locali**: sono le sottocartelle normalmente presenti all'interno di *inetpub*.
- ***directory* virtuali**: definendo una *directory* virtuale *images* stabiliamo che alla **URL** http://Web.aipa.it/images corrisponda una determinata cartella nel disco del nostro ***server***, ad esempio C:\mygifs\. Essendo la cartella su C e non in *Inetpub* non sarebbe stata visibile dal ***server*** *Web*. Per tali cartelle è possibile definire un *alias*, ovvero un nome diverso col quale identificare la cartella su ***Internet***. Per modificare le proprietà di una *directory* virtuale già creata occorre selezionarla con il *mouse* dalla *console* di **IIS** e con il terzo bottone del *mouse* selezionare *Properties*... .
- ***Web***: sono sottocartelle speciali. Oltre a comportarsi come le cartelle virtuali, è possibile associarle ad un utente che, mediante *Front Page* può sincronizzare un proprio sito locale con un *Web*. Le estensioni del ***server*** (*Front Page Server Extension*) gestiscono il meccanismo di sincronizzazione.

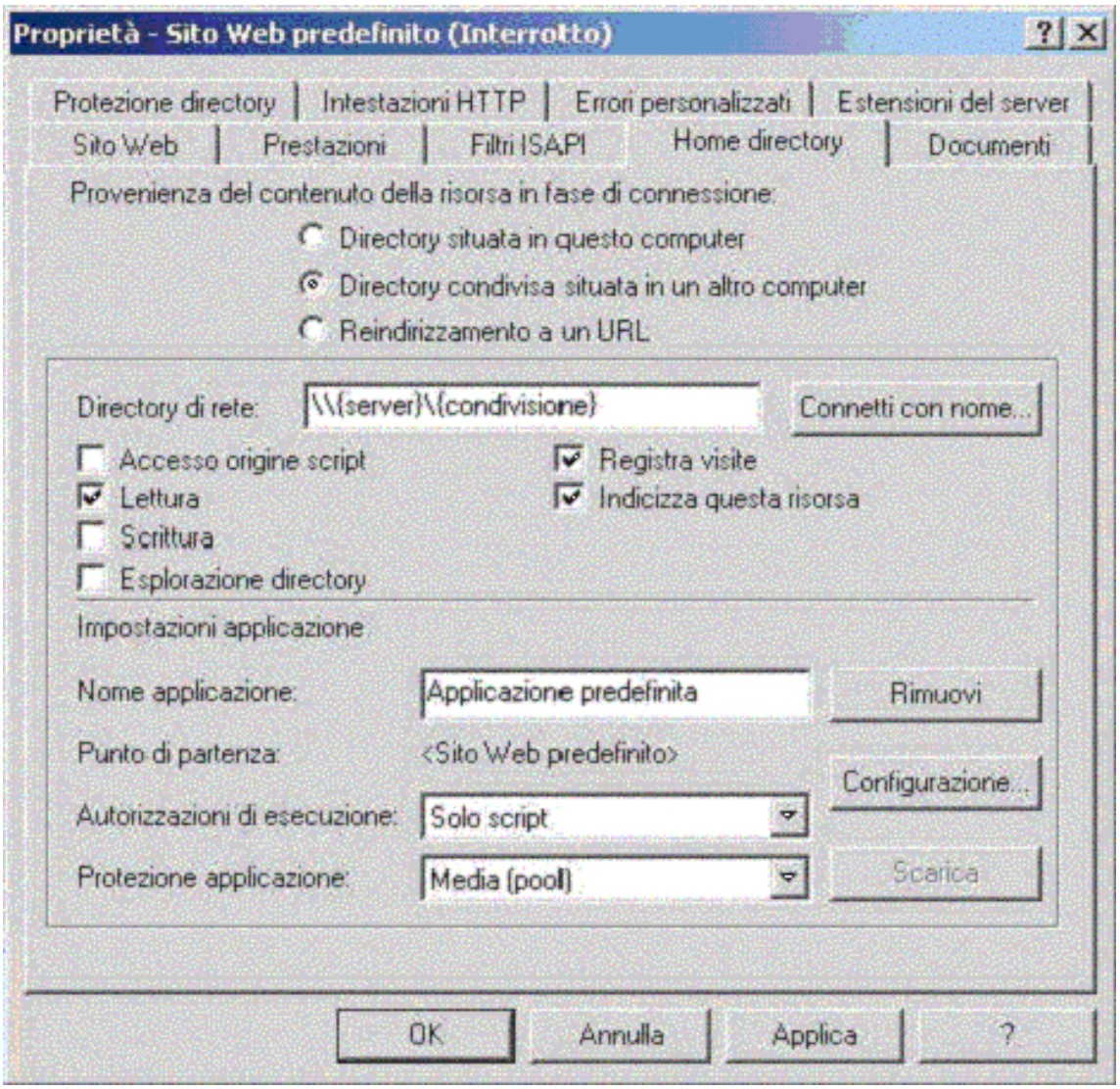


Cartelle del sito Web con i dettagli dei percorsi in IIS

La configurazione del *Web* ***server*** avviene in modo modulare: *click* destro del *mouse* e menù proprietà. Il *click* sulle cartelle permette di configurarne le proprietà. *Click* su tutto il *Web* ***server*** ne consente la configurazione globale.

Impostare le proprietà del server

Nella sezione *Documents* è possibile definire quali sono i nomi del documenti di *default*, ossia da visualizzare quando la **URL** indica la *directory* ma non specifica il documento. Ad esempio se indichiamo la **URL** http://prove.aipa.it/subdir e il documento di *default* è *index.htm* allora il ***server*** restituirà al ***client*** il documento *index.html*. Nel *tab Home directory* invece è possibile impostare il percorso locale da associare al *Web* ***server***, e la modalità alla quale accedere alle cartelle (sola lettura, esplorazione *directory*, eccetera). Le possibilità sono le seguenti:



Finestra delle proprietà di un sito Web IIS, con la specifica della home directory del sito

- Una *directory* sul *filesystem* locale.
- Una *share* (in formato UNC), richiede *User/Password*. Se su dominio diverso entrambi devono avere un utente con lo stesso nome. È sconsigliato.
- Un **URL**.

Nei primi due casi è possibile settare anche:

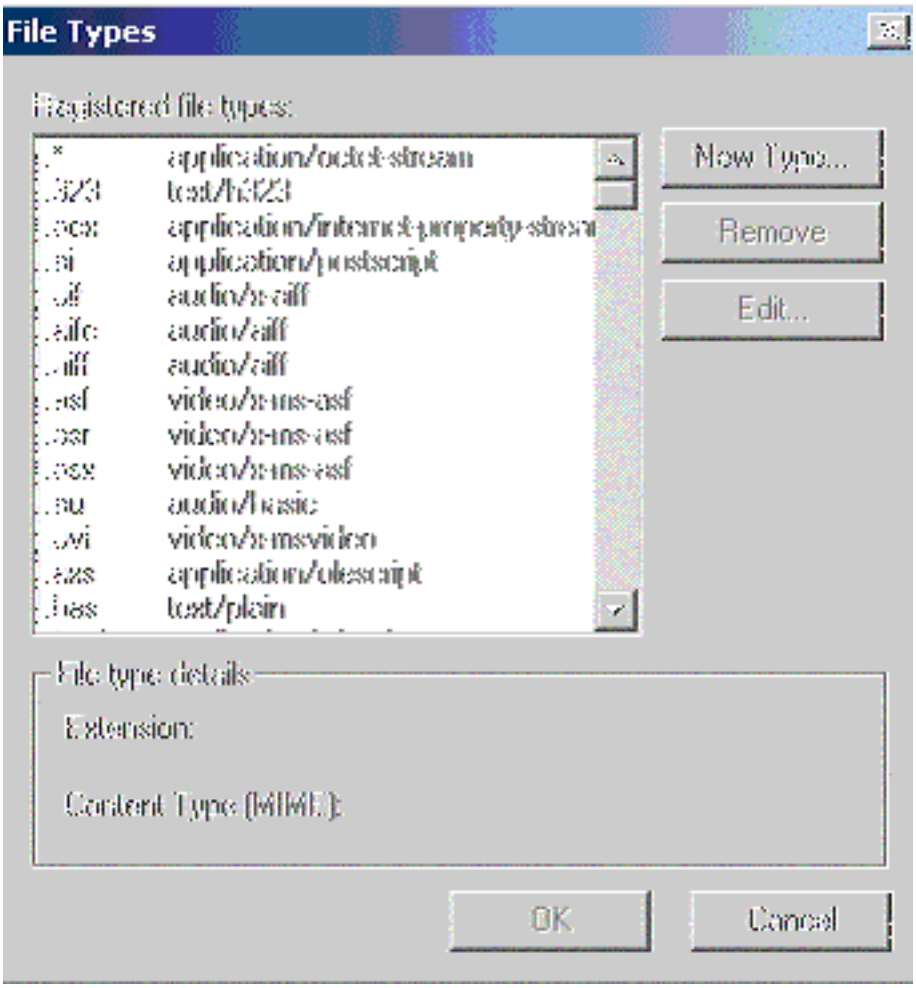
- *Permissions* da applicare. *Read* (*Default*), *Write* (necessita **HTTP** 1.1).
- *Browsing* permesso (in caso di non presenza *default page*) (a livello di sito, non di *directory*).
- *Log*, Sito indicizzato, Sito *FrontPage*.

In **IIS**, una cartella con relativi *file* e sotto-*directory*, viene definita Applicazione. È possibile collegare l'**applicazione** ad una *home page*. Si può far eseguire l'**applicazione** in uno spazio di memoria separato e dare *permission* di esecuzione:

- *None* (non esegue nulla).
- *Script* (esegue solo ***script***).
- *Execute* (esegue ***script*** ed eseguibili NT: dll, exe).

I ***MIME****-Type* invece servono al ***server*** *Web* per spedire al *browser* il tipo del documento. Infatti, secondo il protocollo **HTTP** assieme al documento deve essere spedito anche il tipo di questo documento (**HTML**, PDF, PS, eccetera...). Sapere il tipo del documento serve al *browser* per decidere quale azione intraprendere, cioé se parserizzare il codice e visualizzarlo (**HTML**) oppure aprire una **applicazione** esterna o un *plug-in* (PDF, PS ...). In caso il tipo del *file* ricevuto dal *browser* non sia un ***MIME****-Type* conosciuto il *browser* chiede all'utente cosa fare (salvarlo su disco...).

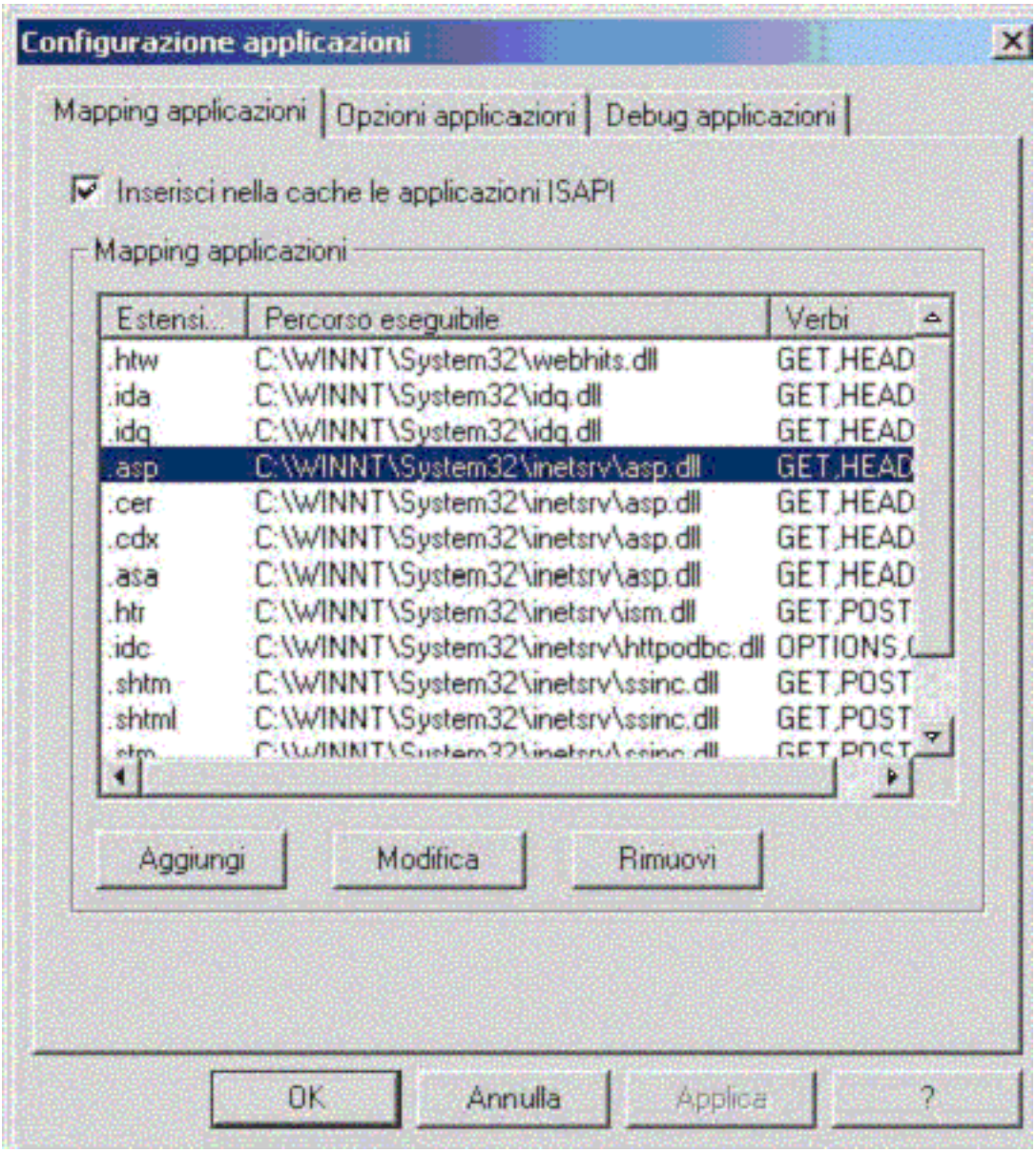
I ***MIME****-Type* del ***server*** *Web* si definiscono cliccando con il bottone destro del *mouse* sul nome del ***server***. Da qui si seleziona *Computer MIME Map* per definirne uno nuovo.



Finestra IIS con il nome dei file ed i loro tipi

La finestra risultante visualizza tutte le estensioni ***MIME*** configurate per quel ***server*** *Web*. Con *New* possiamo definire una nuova estensione dove associamo ad una estensione del *file* un ***MIME****-Type*. Notiamo che i ***MIME*** *type* definiti qui sono validi per tutte le applicazioni del *Web* ***server***.

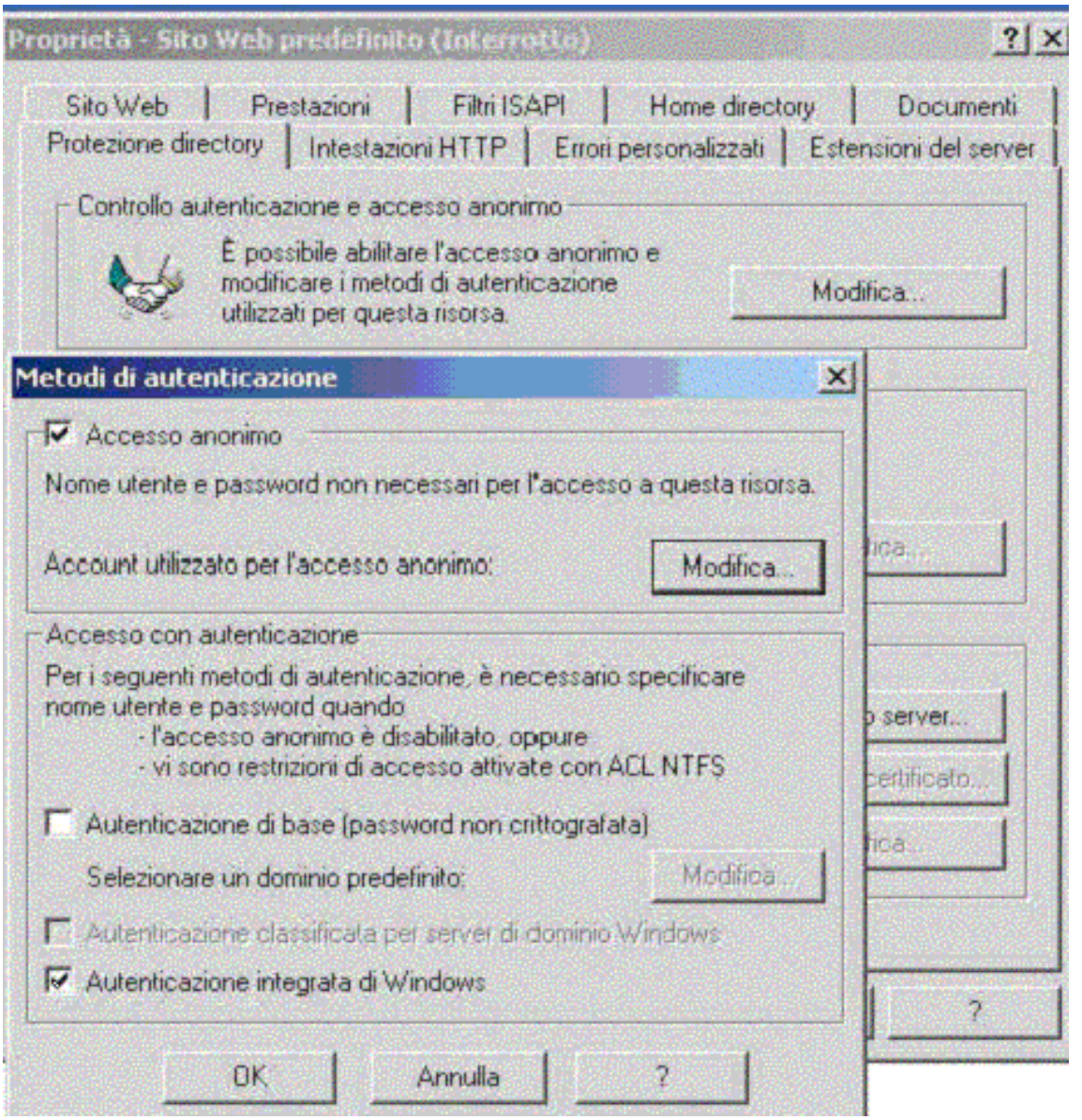
Filtri ISAPI: si possono aggiungere, e quindi gestire filtri. Si possono utilizzare per eseguire applicazioni remote attivate dal tipo di richiesta presente nell'**URL**. Sono DLL attivate dall'estensione dei *file*. Ad esempio richiedere al *Web* ***server*** un *file* **ASP** non solo implica lo scaricamento in locale, ma anche il filtraggio ovvero la computazione attraverso la DLL asp.dll



Finestra IIS delle applicazioni con i percorsi e le operazioni ammesse

Directory Security

Permette di specificare l'*Access Control* alle risorse del *Web* ***server***:



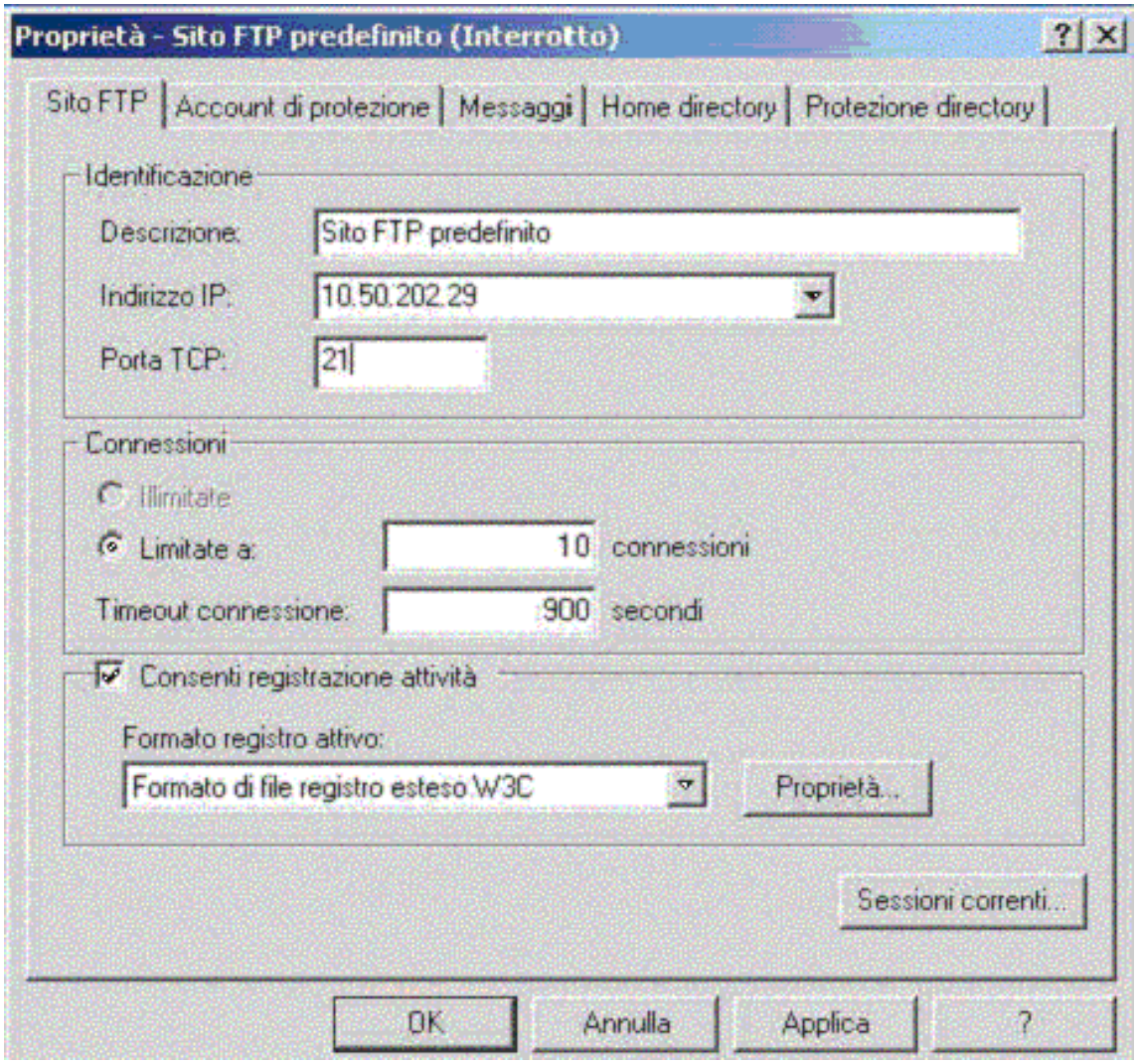
Controllo degli accessi in IIS

Allow ***Anonymous*** *Users* è l'opzione *default* per WWW. Viene comunque utilizzato un utente di NT, colui che fisicamente esegue i processi sulla macchina WWW. *Basic Authentication* richiede utente e *password* in chiaro utilizzando le *permission* NTFS. Richiede di disabilitare la prima opzione e richiede che l'utente esista per l'NTFS. WNT *Challeng/Response*, richiede utente e *password* criptate con l'algoritmo di NT, utilizza le *permission* NTFS e richiede di disabilitare la prima opzione.

Il servizio FTP

È il metodo più usato per trasferire *file* in ***Internet***. È ottimizzato perché utilizza 2 porte, una per mandare e una per ricevere. La connessione tra loro stabilita rimane attiva per tutta la sessione. Utilizza 5 *tab* di proprietà e supporta la stessa gerarchia descritta per il *Web: Master/Default/File*.

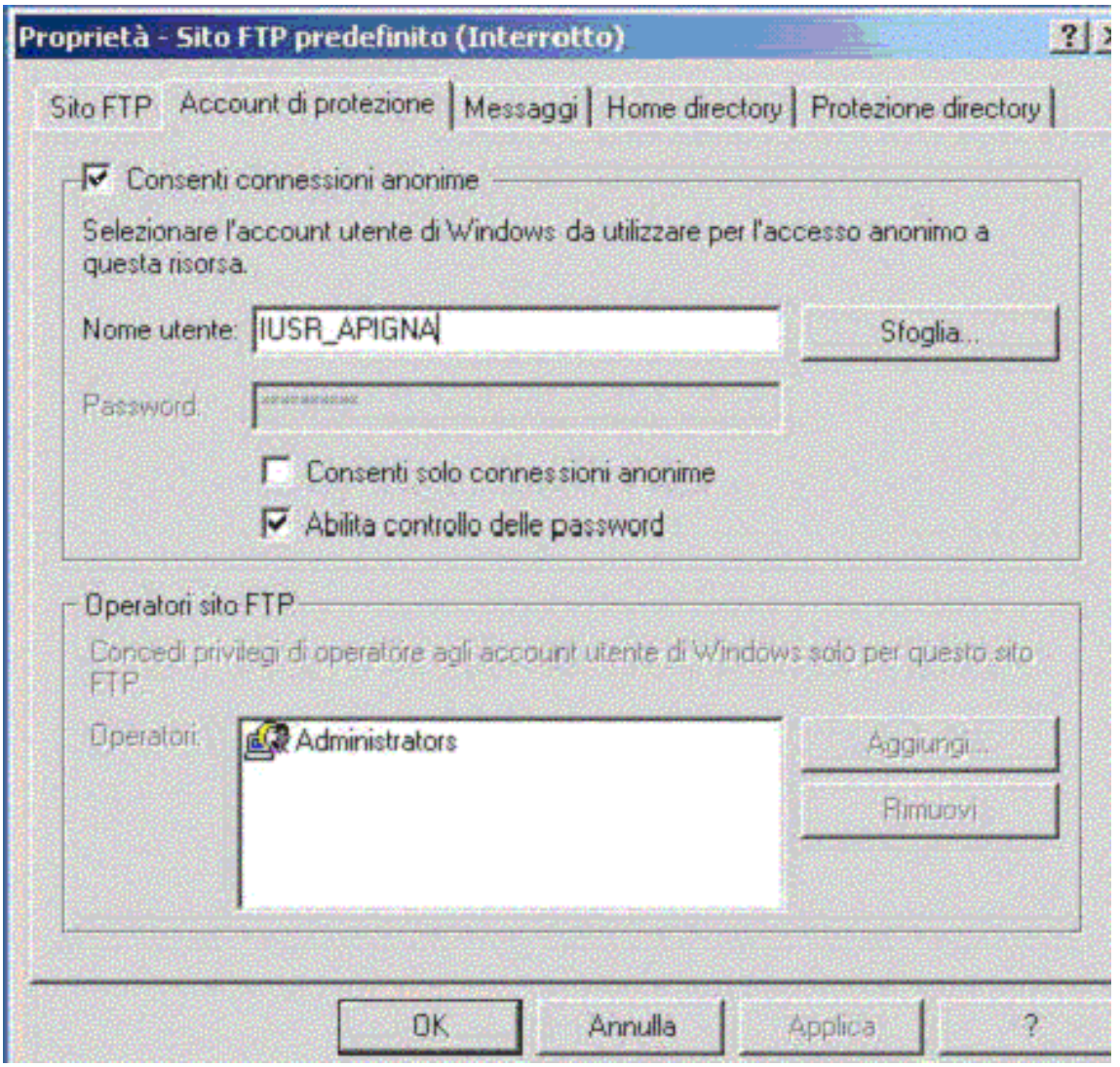
FTP Site: permette di specificare Nome, *IP*, Porta (21), Numero di connessioni massime e *Timeout* di sessione. Con *Current session* è possibile vedere in tempo reale l'elenco degli utenti attualmente collegati al ***server*** **FTP**.



Finestra IIS delle proprietà di un sito FTP

Security Accounts: definisce la sicurezza per il sito **FTP**: con *Allow anonymous account* si dà la possibilità di connessioni anonime, ed occorre specificare l'utente/*password* di NT che verrà utilizzato per validare l'accesso.

Si può specificare di avere SOLO connessioni anonime e di sincronizzare in automatico la *password* di ***Anonymous*** con l'*account* di NT. In *Operators* si può specificare quali *account* di NT possono gestire il sito.



Finestra IIS delle proprietà di un sito FTP con la selezione della cartella 'Account di Protezione'

Messages: si possono impostare dei messaggi da inviare al ***client***. Di *default* sono vuoti: *Welcome*, *Exit* e *Maximum Connections*.

Home directory: può essere locale o remota su un'altro PC. Si può specificare se permettere Lettura, Scrittura o Entrambe e se si vuole tenere il *log* degli accessi a questa *directory*. È possibile inoltre visualizzare la *directory* in stile MS-Dos o *Unix*.

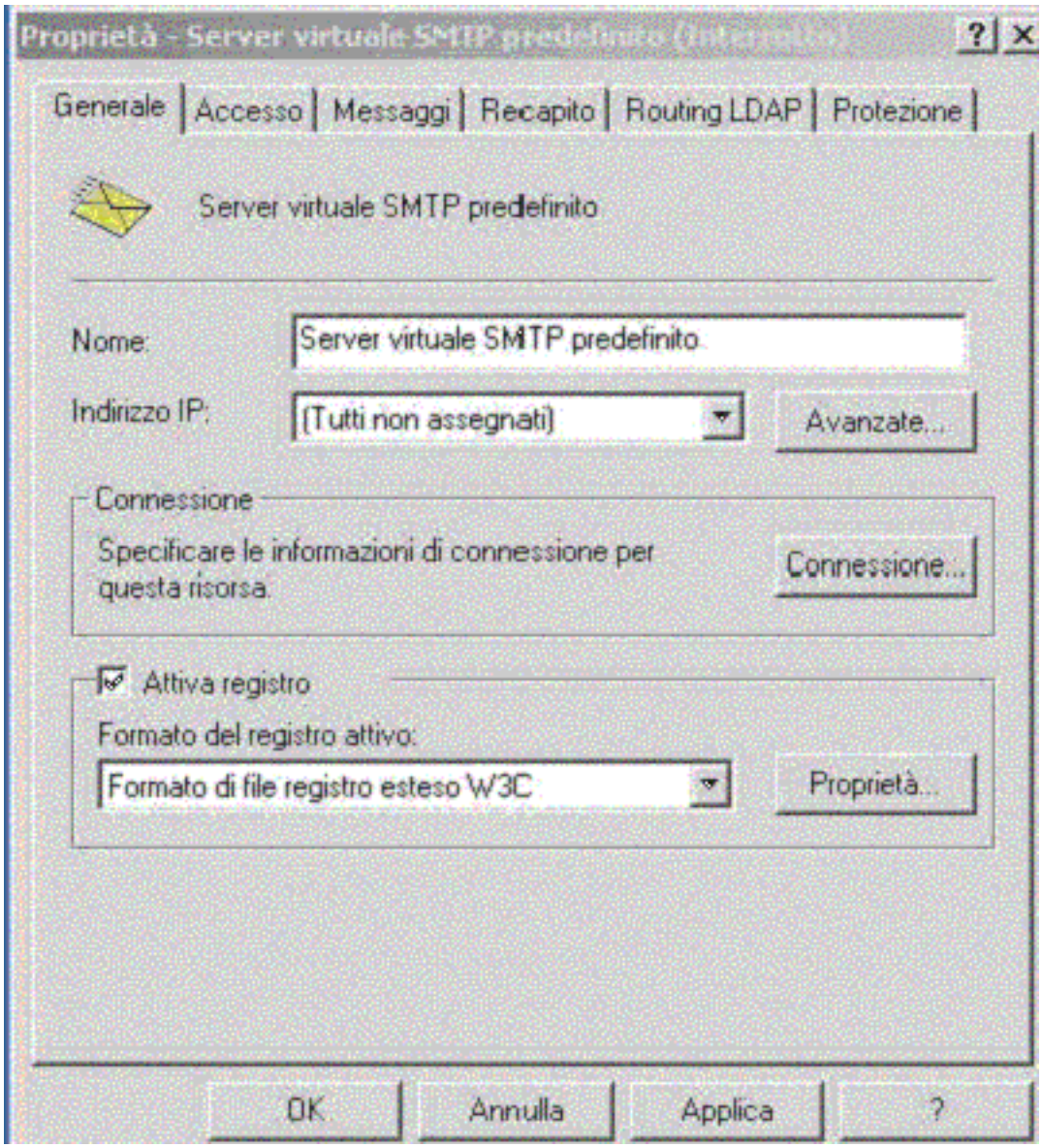
Directory Security: utilizza proprietà ed impostazioni analoghe a quelle descritte per il sito *Web*.

SMTP

Permette di gestire la posta di ***Internet*** ed è amministrabile con MMC o HTMLA. Permette di ricevere tutta la posta in arrivo e metterla in una cartella *Drop*, per ogni Dominio specifico. Per spedire i messaggi utilizza *TCP*. Si possono mettere i messaggi in una cartella *Pickup* in modo da poter essere trasferiti automaticamente.

Per ogni SMTP *Service* è possibile configurare i Domini, e visualizzare le sessioni correnti. Non viene installato di *Default*, ma dall'opzione *custom* su *Windows* NT. È installato per *default* invece su *Windows* 2000 *Server*. All'installazione crea \Inetpub\Mailroot e sotto di essa crea 5 altre cartelle:

- ***BadMail*** Messaggi non recapitabili.
- ***Drop*** Messaggi in arrivo. Si può spostare. Se ne può avere una per dominio.
- ***Pickup*** Messaggi in uscita. Recapitati in automatico.
- ***Queue*** Messaggi in attesa. Vengono depositati quelli che non era possibile inviare.



Finestra IIS delle proprietà di un server SMTP

Funzionamento. Quando un messaggio arriva alla porta *TCP* designata o viene inserito nella *PickUp*, il messaggio viene innanzitutto inserito nella cartella *Queue*. Poi il

server determina se il destinatario è locale o remoto. Se locale sposta il messaggio nella cartella *Drop* del dominio/i configurato, altrimenti viene rispedito.

Modalità di inoltro. I messaggi contenuti nella cartella *Queue* vengono ordinati per dominio e spediti in gruppo. Il ***server*** tenta di contattare il ***server*** remoto per assicurarsi che è disponibile a trasmettere, altrimenti riaccoda il messaggio. Poi vengono verificati i destinatari (se un destinatario non è raggiungibile viene generato un NDR) quindi messaggio viene spedito. Il compito di SMTP *Service* termina nel momento che il ***server*** remoto conferma la ricezione del messaggio. Se è stata abilitata la cifratura **SSL**, il ***server*** cripta i messaggi in uscita.

Attenzione! Mettere in Pausa il servizio SMTP siginifica non accettare connessioni ***client*** ma continuare l'inoltro di posta ai ***server*** remoti.

SMTP *Site*: Permette di specificare nome sito, e *IP address*. Per le connessioni in entrata e in uscita si può configurare separatamente:

- Porta *TCP*. *Default* 25.
- Numero massimo connessioni contemporanee. *Default* 1000.
- *Timeout*. *Default* 600 sec.
- Numero massimo connessioni (in uscita) per dominio. *Default* 100.

Operators: specifica quali utenti di NT hanno diritto ad amministrare il ***server***. Non utilizzabile con HTMLA.

Messages. Si possono impostare le limitazioni sui messaggi da inoltrare. Se un messaggio supera i limiti consentiti viene dichiarato NDR e viene rispedito al mittente. Se neanche il mittente è raggiungibile, viene spostato nella *directory* di *BadMail*. È possibile impostare l'ampiezza del singolo messaggio lato ***server*** e della singola sessione. Se un messaggio in arrivo supera il primo valore viene accettato fino al massimo invalicabile del secondo, dopodiché la sessione viene chiusa. Inoltre si può settare:

- Numero massimo di messaggi da inviare per connessione (20).
- Numero massimo di destinatari per messaggio (100).
- Amministratore che riceverà copia di tutti i messaggi non spedibili, *Directory* di *Badmail*.

Delivery: può essere diviso in 3 categorie di opzioni:

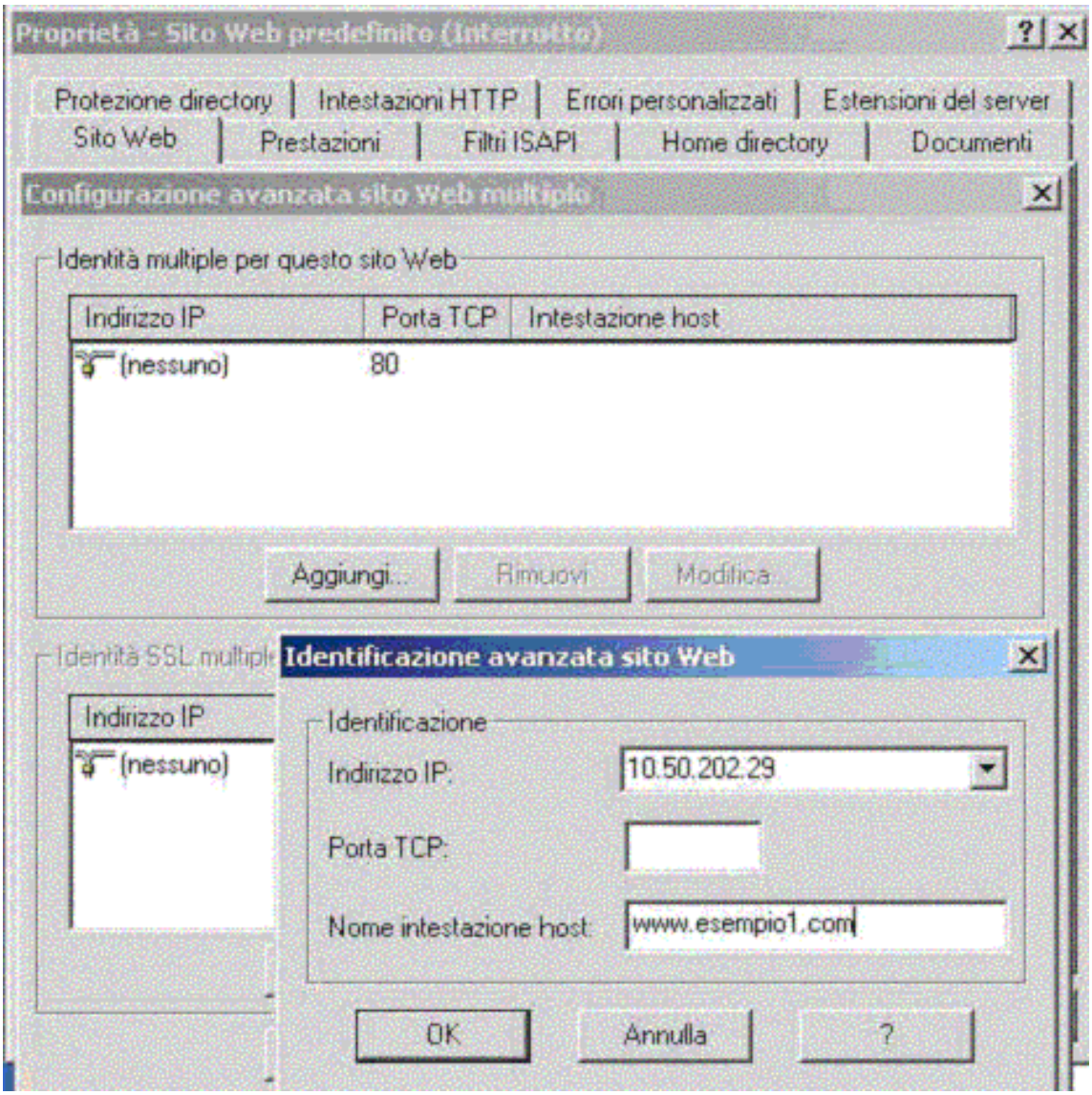
- Trasmissione Numero massimo di tentativi (48) prima di dichiarare un messaggio NDR. Intervallo in minuti tra due tentativi (60) Sono specificati sia per code Locali che Remote. Instradamento Numero massimo di Salti (15) prima di considerare un messaggio NDR.
- Nome del ***server*** di instradamento (MX), Nome del ***server*** *Smart*. Sicurezza *Masquerade Domain*, per nascondere il dominio di provenienza.
- *Reverse* **DNS** dell'*IP* del *sender* per controllare che la *mail* del *From* arrivi effettivamente dal dominio specificato. *Outbound Security* per specificare il tipo di autenticazione supportata per i messaggi in uscita : Nessuna, *Basic*, NT, TLS.

Siti virtuali con intestazioni host

In questo paragrafo viene illustrata la procedura dettagliata per l'*hosting* di più siti *Web* utilizzando un unico **indirizzo** *IP*. *Microsoft Internet Information Services* (**IIS**) consente di eseguire il *mapping* di più siti *Web* aventi lo stesso numero di porta a un unico **indirizzo** *IP* utilizzando una funzionalità denominata Nome intestazione ***host***. Attraverso l'assegnazione di un nome intestazione ***host*** univoco a ciascun sito *Web*, questa funzionalità consente di eseguire il *mapping* di più siti *Web* a un solo **indirizzo** *IP*.

Per configurare siti *Web* utilizzando la funzionalità Nome intestazione ***host***, eseguire le seguenti operazioni:

- Fare *click* con il pulsante destro del *mouse* sul sito *Web* desiderato, quindi scegliere Proprietà dal menù di scelta rapida.
- Nel gruppo **Identificazione** sito *Web* selezionare l'**indirizzo** *IP* che si desidera assegnare al sito *Web* nell'elenco Indirizzo *IP*.
- Fare *click* sul pulsante Avanzate.
- Nel gruppo Identità multiple per questo sito *Web* fare *click* sull'**indirizzo** *IP*, quindi scegliere modifica. Verrà visualizzata la finestra di dialogo Identificazione avanzata sito *Web*.
- Nella casella Nome intestazione ***host*** digitare l'intestazione ***host*** desiderata. Ad esempio, digitare www.esempio1.com. Aggiungere il numero di porta, selezionare l'**indirizzo** *IP* dall'elenco, quindi scegliere OK.



Finestra IIS per la configurazione di un sito Web multiplo

Se si desidera configurare il sito *Web* con identità aggiuntive, scegliere Aggiungi. Utilizzare lo stesso **indirizzo** *IP* e porta *TCP*, ma immettere un Nome intestazione ***host*** univoco. Se si desidera ad esempio accedere allo stesso sito *Web* sia da ***Internet*** che da una rete *Intranet* locale, è possibile configurare l'identità del sito *Web* come indicato di seguito:

192.168.0.100 80 www.esempio1.com
192.168.0.100 80 esempio1.com

- Fare *click* con il pulsante destro del *mouse* sul sito *Web* successivo, quindi scegliere Proprietà dal menù di scelta rapida.
- Nell'elenco Indirizzo *IP* selezionare lo stesso **indirizzo** *IP* selezionato al passaggio 4, quindi scegliere Avanzate.
- Nel gruppo Identità multiple per questo sito *Web* fare *click* sull'**indirizzo** *IP*, quindi scegliere Modifica. Verrà visualizzata la finestra di dialogo

Identificazione avanzata sito *Web*.

- Nella casella Nome intestazione ***host*** digitare un'intestazione ***host*** univoca per il sito *Web*. Ad esempio, digitare www.esempio2.com. Aggiungere il numero di porta, selezionare l'**indirizzo** *IP* dall'elenco, quindi scegliere OK.
- Ripetere i passaggi illustrati per tutti i siti *Web* che si desidera ospitare nello stesso **indirizzo** *IP*.
- Registrare le intestazioni ***host*** con il sistema di risoluzione dei nomi appropriato, ad esempio un ***server*** **DNS** (*Domain Name System*) oppure, per una rete di piccole dimensioni, un *file Host*.

I siti *Web* sono ora configurati in modo tale da accettare richieste *Web* in ingresso sulla base delle rispettive intestazioni ***host***.

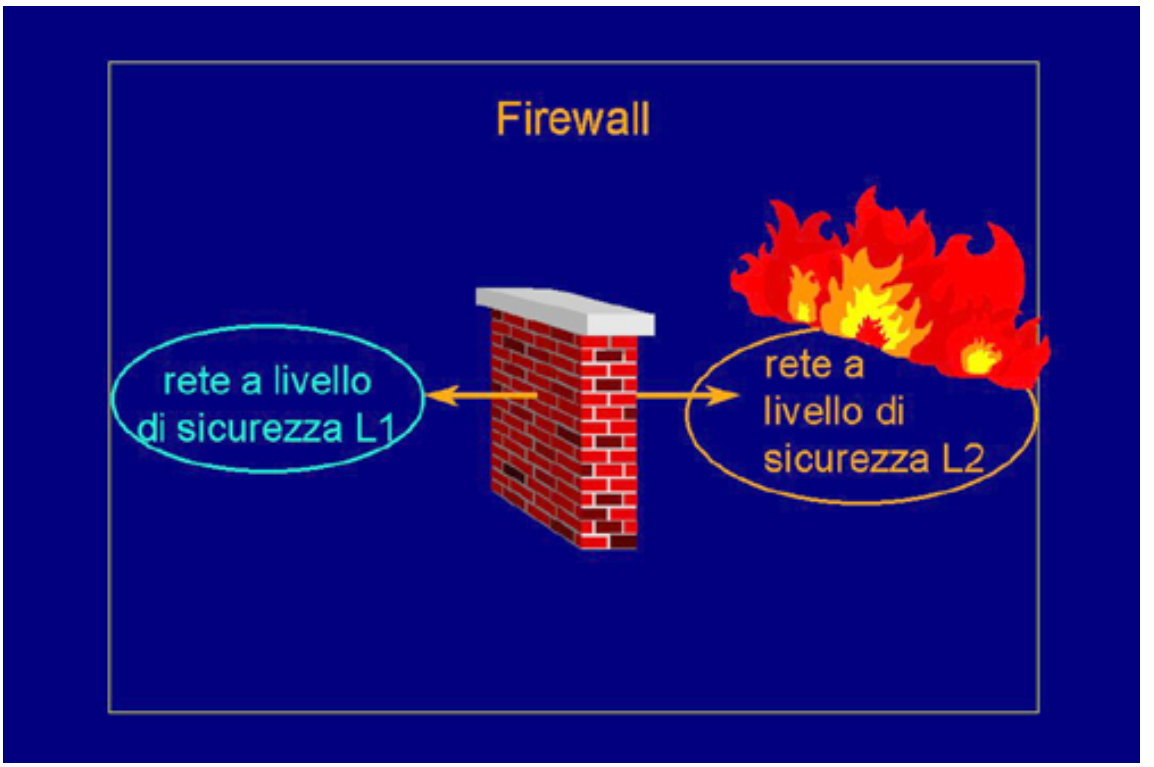
Attenzione, non assegnare un'intestazione ***host*** al sito *Web* predefinito. Molti programmi prevedono che il sito *Web* predefinito utilizzi un **indirizzo** *IP* configurato su (Nessuno), una porta *TCP* configurata su 80 e non utilizzino alcun nome intestazione ***host***.

Autenticazione ed integrità dei dati

Cosimo Laneve

17.2.1 (Spiegare i motivi per cui è necessario un firewall e le sue funzioni)

Firewall



Firewall

Per proteggere una rete dagli attacchi provenienti dall'esterno si utilizza normalmente un sistema denominato *Firewall*. *Firewall* è un termine inglese che indica i muretti di mattoni che, tipicamente, vengono frapposti fra le case americane. Poiché le case americane sono normalmente costruite in legno, si vuole evitare che, se una casa prende fuoco, quindi c'è un incendio, questo si trasferisca automaticamente anche alla casa affianco. Perciò gli americani fanno le case in legno e i muretti di divisione tra una casa e l'altra in mattoni. Questo concetto è stato esteso anche alla sicurezza informatica: ossia nell'ipotesi che esista una rete con un livello di sicurezza L1 e una rete con un altro livello di sicurezza L2, inferiore, quindi più facilmente attaccabile, più facilmente bruciabile e che quindi può essere attaccata più facilmente, il ***firewall*** consiste in una sorta di muretto informatico che deve evitare che il fuoco si propaghi. Il ***firewall*** deve permettere la comunicazione fra le due reti solo, ed esclusivamente, se in quella esterna non c'è il fuoco, mentre deve bloccare automaticamente la comunicazione in caso di tentativi di attacco.

I tre comandamenti dei firewall

I tre comandamenti dei firewall

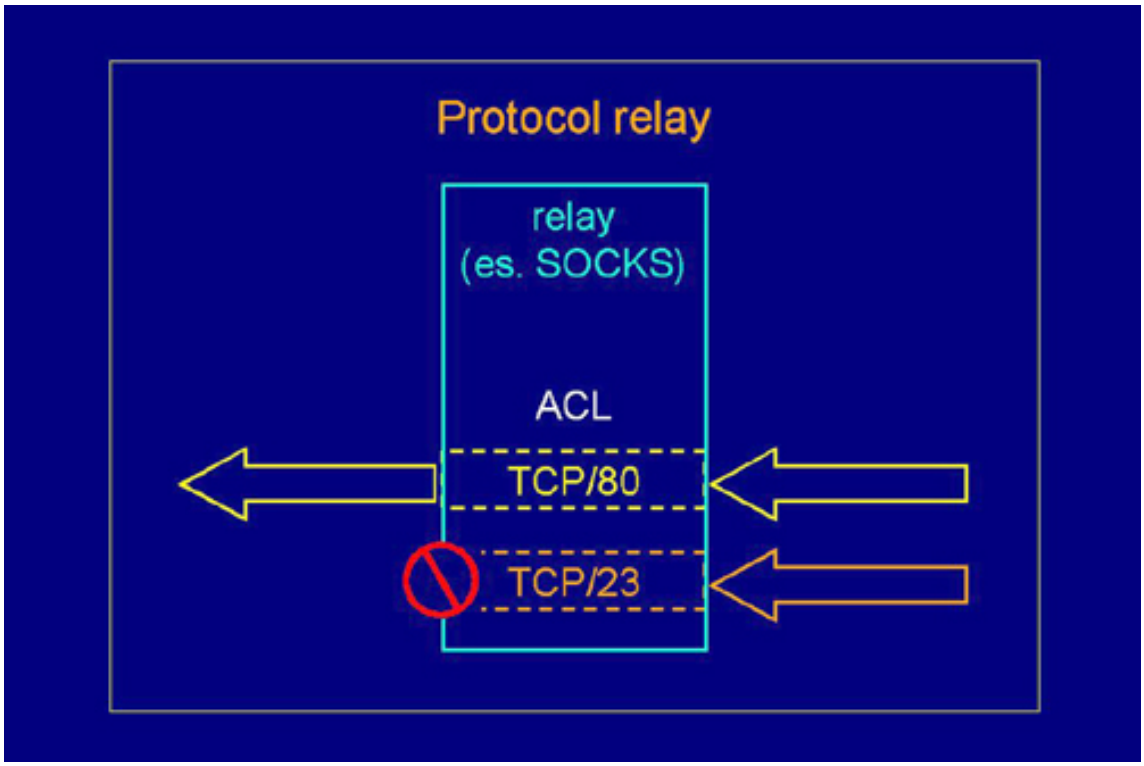
Esistono tre principi fondamentali per costruire un buon ***firewall***. Il primo principio dice che il ***firewall*** deve essere l'unico punto di contatto tra le due reti a diverso livello di sicurezza. Questo è un punto fondamentale. Molto spesso capita che nelle aziende, nelle organizzazioni, esista un ***firewall*** che protegge il collegamento principale fra la rete aziendale e *Internet*, ma poi esistono dei collegamenti secondari che non sono protetti; esistono, ad esempio, dei modem collocati direttamente negli uffici per fare funzioni ausiliarie, a volte anche soltanto per comodità degli utenti. Questa è una cosa assolutamente sbagliata e contraria a tutti i principi. Equivale ad aver blindato la porta di casa e aver lasciato una semplice zanzariera sul giardino: è chiaro che i ladri tenteranno di entrare non dalla porta più robusta, ma da quella più debole. Quindi si ribadisce il concetto: il ***firewall*** deve essere l'unico punto di contatto tra la rete da proteggere e la rete esterna. Un secondo principio inderogabile dice che soltanto il traffico autorizzato può attraversare il ***firewall***. Si noti che ho messo autorizzato fra virgolette, per evidenziare il fatto che nel caso in cui manchi una politica di autorizzazione, ossia non sia stato definito quali sono i tipi di pacchetti, i tipi di protocolli, le operazioni lecite tra la rete interna e la rete esterna, non è possibile creare un buon ***firewall***. Come spesso capita, ubbidire alle direttive che arrivano dall'alto, che dicono: comprate un ***firewall*** perché in questo modo avremmo fatto sicurezza, non aiuta assolutamente, anzi, genera confusione se prima non è stata fatta una analisi di quali sono i requisiti di sicurezza, quindi non sono stati definiti quali sono i tipi di traffico autorizzati ad attraversare il ***firewall***. Infine, il terzo punto dice che un ***firewall*** deve essere un sistema sicuro esso stesso. Quindi, deve essere implementato tramite una serie di sistemi protetti, dei sistemi che sono diversi dai normali *router*,o nodi di elaborazione, nelle configurazioni standard, perché queste configurazioni standard sono normalmente attaccabili e, ovviamente, costruire un castello con dei massi che si sbriciolano è una pessima idea.

Packet filter

Packet filter

Esaminiamo quindi i principali tipi di componenti che vengono utilizzati per creare il sistema ***firewall***. Il primo tipo di componente che esaminiamo è un cosiddetto *packet filter*. Il *packet filter* è una sorta di setaccio che ha il compito di scremare i pacchetti, ossia di buttare via, non lasciar transitare, i pacchetti che hanno delle caratteristiche sconvenienti, non aderenti la nostra politica di sicurezza. Si parla di *packet filter* quando si vanno ad esaminare gli indirizzi di rete, ad esempio l'**indirizzo** sorgente e l'**indirizzo** destinazione, o quando si vanno ad esaminare i protocolli e le porte, ossia quando, in generale, si effettua un filtraggio a livello 3 o 4, eventualmente anche a livello 2. Un *packet filter* può essere implementato tramite apparecchiature *hardware*, ad esempio un *router* sulle quali siano state attivate le **ACL** (liste di controllo degli accessi), oppure può essere realizzato un *software*, tramite un *gateway* su un **nodo** di elaborazione dotato, ad esempio, di due interfacce di rete. Il *packet filter* è un componente molto utilizzato all'interno dei ***firewall***, ma deve essere molto chiaro che, trattandosi di un controllo effettuato a livello 3 o 4, si tratta di un controllo poco raffinato. Quindi, il compito di un *packet filter* è fare un filtro a grana grossa, scremare il grosso dei pacchetti, toglierci dai piedi gli attacchi più banali. Ben difficilmente un *packet filter* potrà essere in grado di andare ad identificare gli utenti, o le applicazioni, o i comandi, che noi vogliamo far passare o controllare attraverso il ***firewall***.

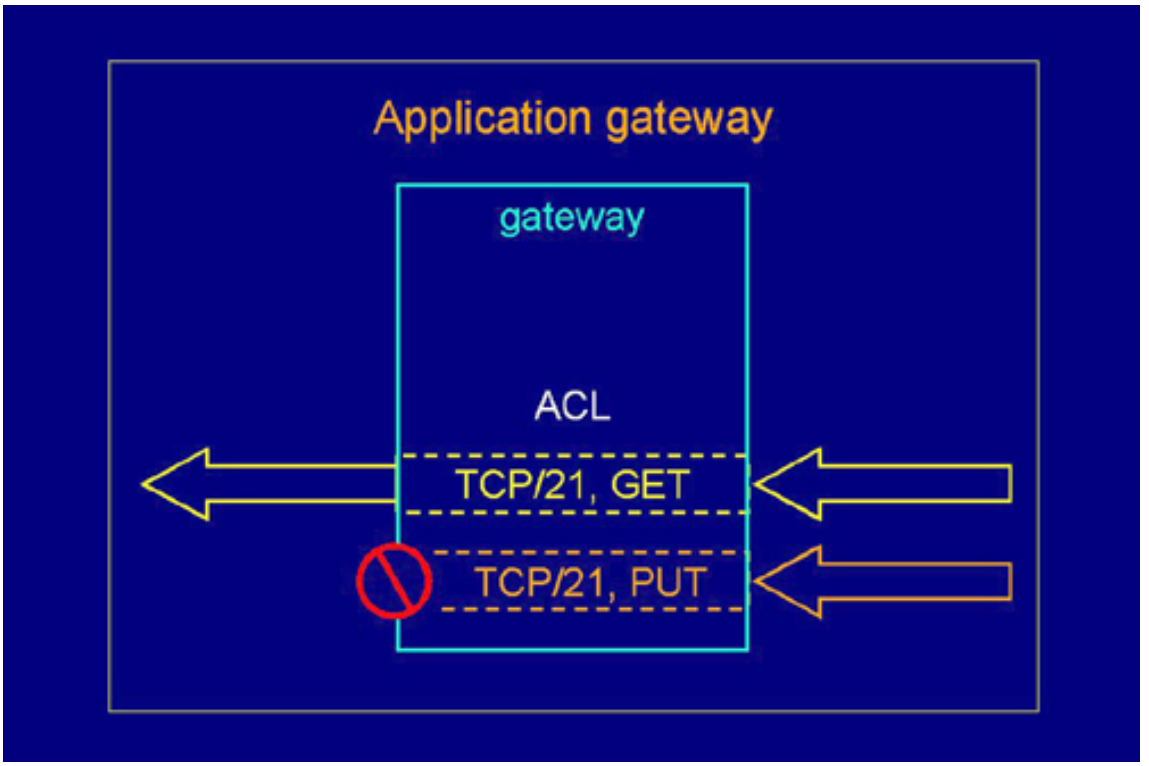
Protocol relay



Protocol relay

Un secondo componente talvolta usato nella creazione di un sistema ***firewall*** è il *protocol relay*. Il *protocol relay* è tipicamente un *software* dotato di una lista di controllo degli accessi basata sui livelli tre e quattro, in particolare sul livello 4. Ad esempio, supponiamo che arrivi una richiesta di collegamento destinata ad un **nodo** *Web* esterno, ossia un **nodo** con protocollo *TCP* porta 80. Se l'**ACL** prevede la possibilità per gli utenti interni di collegarsi a *Web* esterni, questo pacchetto verrà lasciato transitare. Supponiamo però che arrivi un altro tipo di pacchetto, un altro tipo di richiesta di apertura di un canale logico, in questo caso si richiede l'apertura di un canale *TCP* destinato alla porta 23 di un **nodo** esterno. Questa è una richiesta per un collegamento in emulazione di terminale secondo il protocollo telnet. Supponendo che la **ACL** non permetta questo tipo di collegamento: sarà compito del *protocol relay* impedire il collegamento. Nel caso precedente in cui invece il collegamento era permesso, il compito del *protocol relay* è quello di accettare l'apertura di questo canale, aprire un altro canale esterno e far transitare automaticamente i dati fra i due canali. Uno dei sistemi più utilizzati per realizzare un *protocol relay* è il sistema chiamato *SOCKS*, che tra l'altro è un sistema di *public domain*.

Application gateway



Application gateway

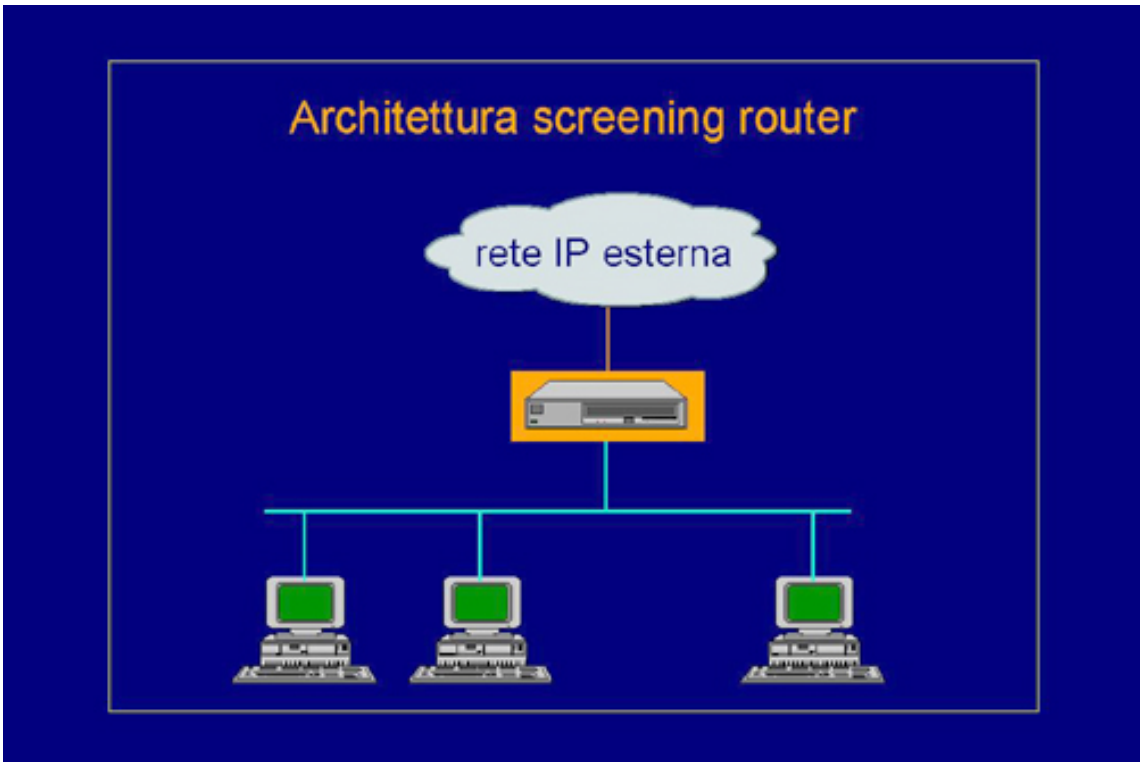
Se vogliamo effettuare dei controlli più raffinati, che non siano solo a livello 3 o a livello 4, dobbiamo salire fino a livello 7. A livello 7 abbiamo a disposizione come sistemi di sicurezza per un ***firewall*** i cosiddetti *application gateway*. Un *application gateway* è un punto di controllo dotato anch'esso di una **ACL**, che è però in grado di andare ad effettuare i controlli in base ai dati, o ai comandi applicativi, contenuti nel *payload* di livello 7. Ad esempio, supponiamo di ricevere una richiesta di transito, attraverso il ***firewall***, relativa all'apertura di un canale *TCP*, porta 21. Questo ci indica che si tratta di un trasferimento di *file* secondo il protocollo **FTP**, ma in particolare l'utente richiede un operazione di ***GET***. Questa operazione di ***GET*** è visibile perché noi siamo un *application gateway* e non siamo un semplice *protocol relay* o un *packet filter*. Se l'**ACL** permette che gli utenti interni prendano dei documenti dall'esterno, lascerà transitare questo comando e quindi l'utente potrà ottenere il documento desiderato. Se però una successiva richiesta evidenzia sempre un trasferimento di dati del protocollo **FTP**, ma in cui la direzionalità questa volta è quella determinata dal comando *PUT*, ossia l'utente interno desidera mandare fuori un documento, la nostra **ACL** potrebbe in questo caso vietare il trasferimento. Si noti che la decisione se far transitare i dati oppure no è stata presa in base al comando a livello applicativo: ***GET*** oppure *PUT*. Questo è un tipo di funzionalità che non poteva in nessun modo essere assolta né da un *packet filter*, né da un *protocol relay*.

Bastion host

Bastion host

Abbiamo detto che un sistema ***firewall*** deve essere dotato di misure di sicurezza interne, esso stesso. In questo senso normalmente si parla di *bastion host*. Un *bastion host* è un sistema fortificato, o come talvolta si suol dire blindato, questo significa che si tratta di un normale sistema operativo e di un normale *hardware*, a cui sono state assegnate funzioni di controllo o di filtraggio. Ad esempio, su questa macchina può essere ospitato un *gateway* a livello 4 o a livello 7. Ma soprattutto sul *bastion host* è stata fatta una configurazione particolare. Ad esempio, è stato installato solo ed esclusivamente il *software* indispensabile per le funzionalità di sicurezza. Analogamente, sono stati installati solo i servizi indispensabili. Bisogna ricordarsi che qualunque **processo** presente su uno dei nodi che costituiscono il ***firewall***, costituisce un potenziale punto di attacco per gli *hacker*. Ecco quindi che bisogna cercare di minimizzare il *software* sia installato sia effettivamente attivo sul ***firewall***, per minimizzare le possibilità di attacco dall'esterno. Inoltre, sul nostro *bastion host* bisognerà cercare di non avere nessun utente: la cosa ideale sarebbe gestire le macchine che compongono il ***firewall*** direttamente dalla loro *console*. Sono ammesse gestioni remote solo ed esclusivamente attraverso canali molto forti, ben autenticati con crittografia e protezione di integrità dei dati. Siccome non esiste la certezza di aver configurato in modalità effettivamente molto sicura un qualunque **nodo** di elaborazione, neanche il miglior esperto di sicurezza potrà mai garantirlo, ecco che un sistema fortificato deve fare un *log* molto estensivo di tutto ciò che capita sul sistema stesso. In questo caso il *log* ci serve per verificare che effettivamente il sistema non sia stato attaccato. Poiché i *log* possono diventare molto grossi, è molto importante che sul sistema siano attivati anche dei sistemi di allarme automatico, ossia dei sistemi che controllano i dati contenuti nei *file* di *log* e evidenzino automaticamente, lanciando degli allarmi, se il sistema ***firewall*** è sotto attacco.

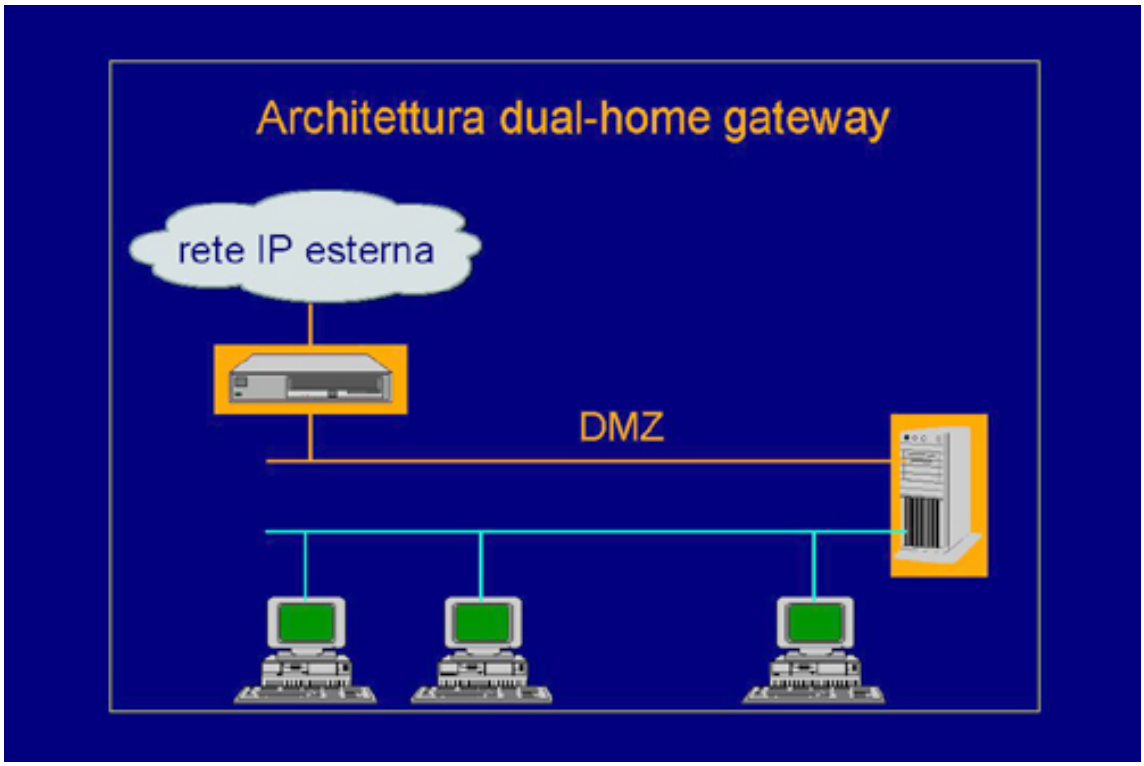
Architettura screening router



Architettura screening router

A questo punto siamo in grado di andare a delineare alcune delle tipiche architetture di ***firewall*** che vengono utilizzate, componendo insieme i vari sistemi base che abbiamo visto fino adesso. L'architettura cosiddetta *screening router* è quella che protegge un'intera rete, nei confronti delle reti esterne, solo ed esclusivamente tramite il *router*. Ossia, il *router*, che già avrebbe il compito di instradare i pacchetti tra la nostra rete e la rete esterna, ha un compito supplementare: quello di effettuare dei filtraggi, ovviamente al livello che gli è possibile, ossia tipicamente ai livelli 3 e 4. A questo punto tutta la sicurezza del nostro sistema è demandata al *router* stesso: se il *router* effettua un buon lavoro saremo sicuri, se non effettua un buon lavoro, come purtroppo non può fare visto che non può salire fino ai livelli applicativi, la nostra rete sarà esposta. L'architettura *screening router*, quindi, si presta ad essere utilizzata, come soluzione di ***firewall***, solo ed esclusivamente in casi di reti con pochissimi protocolli che transitano tra la rete e l'esterno e soltanto per reti con livello di sicurezza medio-basso.

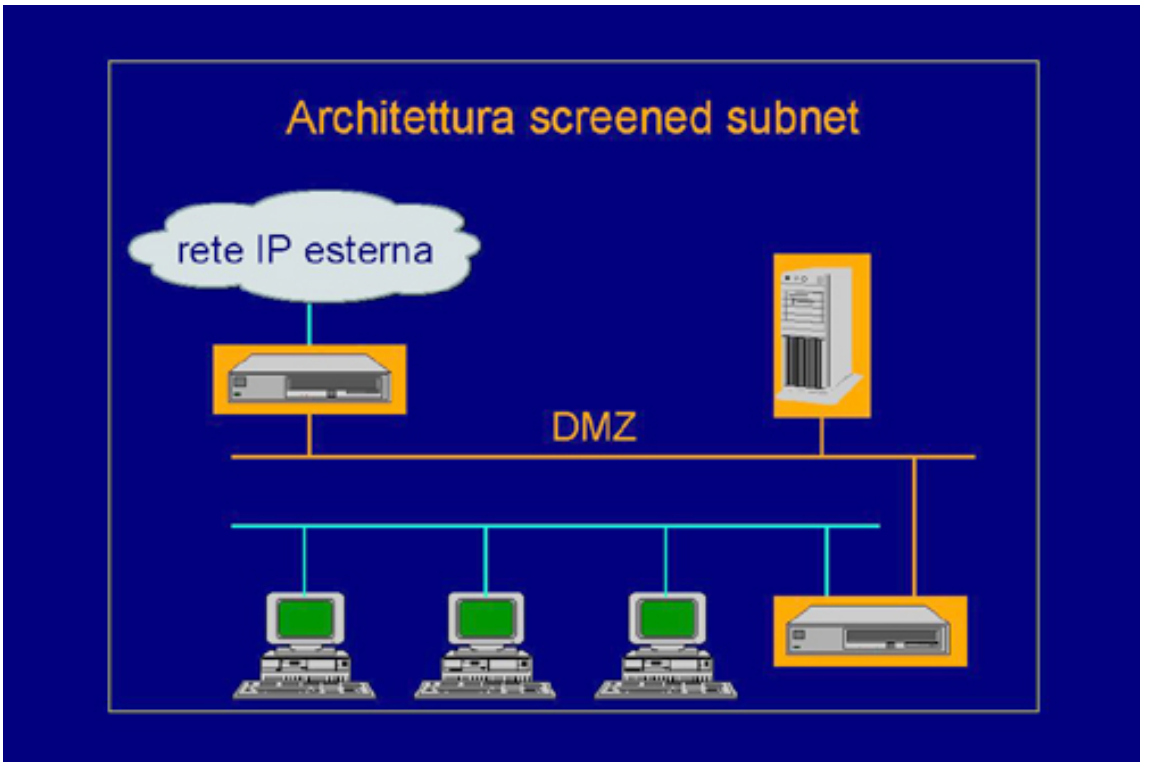
Architettura dual-home gateway



Architettura dual-home gateway

Se si desidera migliorare il livello di sicurezza offerto da un'architettura di quel genere, conviene introdurre un secondo elemento in grado di effettuare dei controlli ad un livello più elevato. Si parla, allora, di un'architettura di tipo *dual-home gateway* nel caso in cui, oltre al *router*, il controllo venga effettuato anche da un **nodo** di elaborazione dotato di due schede di rete. Questo **nodo** di elaborazione separerà, obbligatoriamente, la rete interna, non soltanto da una rete esterna, ma anche da una rete intermedia. Questa rete intermedia prende il nome di zona demilitarizzata, ossia terra di nessuno, perché è quella entro cui dovrebbero passare solo un sottoinsieme dei pacchetti provenienti dall'esterno. Il *router* fa un primo filtro a livello basso, 3 e 4, il *gateway* effettua un controllo più raffinato a livello 5, 6 o 7. Sulla zona demilitarizzata sono normalmente ospitati quei ***server*** che hanno necessità di frequenti contatti con l'esterno. Ad esempio, il ***server*** *Web* aziendale rivolto all'esterno troverebbe logicamente posto qui, all'interno della DMZ.

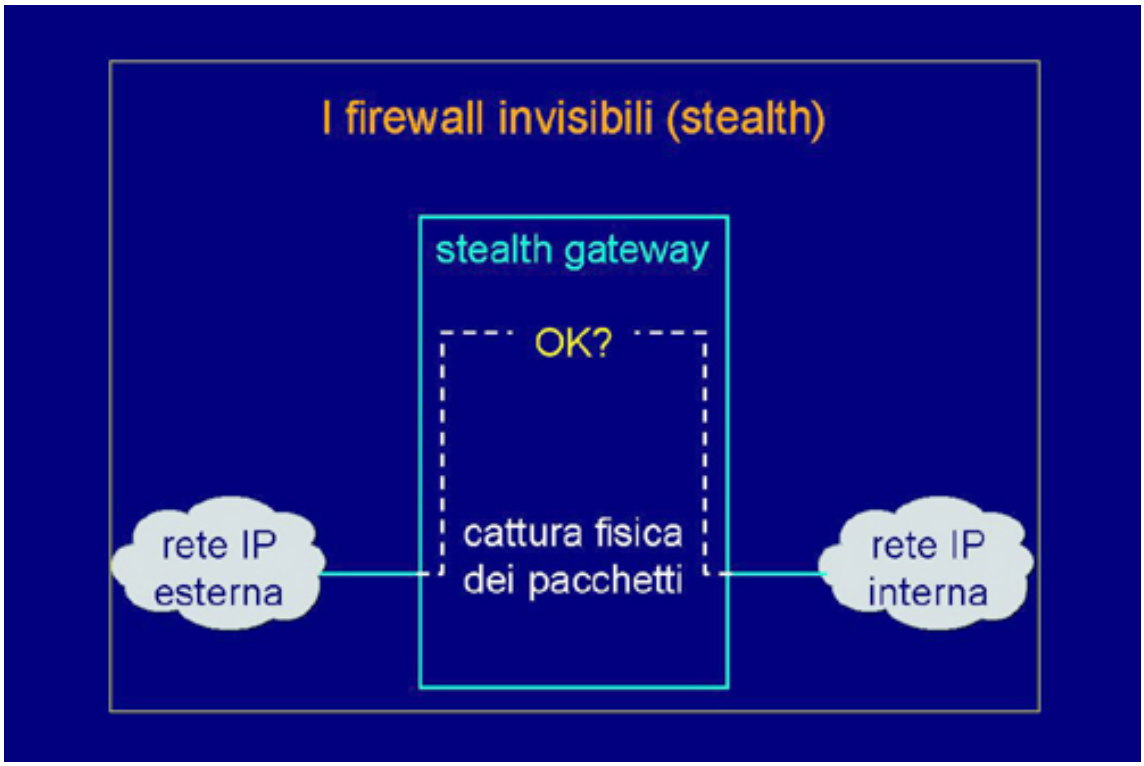
Architettura screened subnet



Architettura screened subnet

Infine, l'architettura migliore è quella cosiddetta *screened subnet*. In un architettura di tipo *screened subnet* noi siamo in grado di nascondere completamente l'esistenza della rete interna, al mondo esterno. Questo è possibile perché, rispetto alla soluzione precedente, la zona demilitarizzata è stata completamente nascosta tramite un secondo *router*. In questo modo il primo *router* ignorerà completamente l'esistenza della rete interna. Un eventuale attaccante dovrà in ogni caso superare almeno due sistemi, prima di penetrare all'interno della nostra rete aziendale. Anche in questo caso, questa rete su cui non sono attestati nodi aziendali, si chiama zona demilitarizzata ed analogamente qua sopra vengono attestati i ***server*** esterni. Notate che nel caso che il nostro sistema informatico preveda la disponibilità di modem collegati direttamente alla nostra rete aziendale, tali modem e il relativo NAS (*Network Access Server*) devono essere posizionati sulla zona demilitarizzata. In nessun caso bisogna permettere che delle apparecchiature di collegamento insicure, quali sono i modem, siano collegati direttamente alla rete aziendale.

I firewall invisibili (stealth)



I firewall invisibili (stealth)

Infine, concludiamo con un concetto innovativo ed interessante, che è emerso in anni recenti nel campo dei ***firewall***. Uno dei problemi dei ***firewall*** è che essi stessi potrebbero essere oggetto di attacco da parte di un *hacker*. Ma un *hacker*, per attaccare un sistema, ha bisogno di potergli indirizzare dei pacchetti. Il concetto che è stato sviluppato è quello di ***firewall*** invisibile, ossia un ***firewall***, quindi un **nodo** di elaborazione, privo di **indirizzo** di rete. Se un **nodo** non è dotato di **indirizzo** di rete non gli si possono indirizzare dei pacchetti e non è quindi virtualmente attaccabile. Ma allora come riesce a sviluppare le proprie funzionalità di sicurezza? Il concetto è molto semplice, il ***firewall*** invisibile, il cosiddetto *stealth* ***firewall*** o *stealth gateway*, è costituito da un **nodo** di elaborazione che interrompe fisicamente il cavo tra la rete interna e la rete esterna. Dopodiché effettua cattura fisica dei pacchetti, perché nessuna delle sue interfacce è dotata di **indirizzo** di rete, ma grazie al fatto che il cavo entra fisicamente dentro l'interfaccia è possibile catturare tutti i bit. Questi bit sono sottoposti a un controllo di livello opportuno, secondo le funzionalità di sicurezza richieste e se il controllo da esito positivo i bit vengono inoltrati inalterati. Quindi, in nessun modo avviene una modifica dei dati: un osservatore esterno non può in nessun modo accorgersi del fatto che i dati sono stati controllati. I dati subiscono soltanto un lieve ritardo nel momento in cui attraversano il *gateway*. L'azione del *gateway* a tutti gli effetti è invisibile, i pacchetti passano o non passano, sembra in modo magico. Questo è un concetto interessante che mi aspetto di vedere applicato sempre più estesamente in futuro.

Ottimizzazioni delle prestazioni di un Web server

Cosimo Laneve

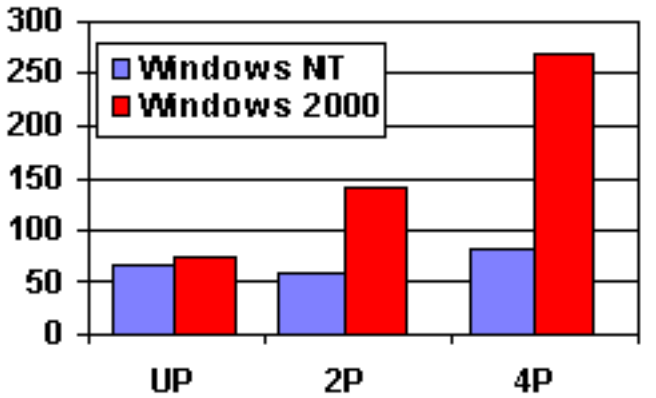
17.1.1 (Installare e configurare un Web server)

Ottimizzazioni delle prestazioni di un Web server

Spesso il ***server*** non è in grado di gestire tutto il carico di cui è gravato. Inoltre, una volta messi in produzione, i ***server*** devono avere la possibilità di crescere: un ***server*** può avere inizialmente 100 utenti, ma potrebbe passare a 500 utenti nel giro di poco tempo.

Secondo una definizione generica, il *benchmarking* è il **processo** che raffronta due o più sistemi (o componenti di un sistema) al fine di stabilire quale sia in grado di fornire le migliori prestazioni o di consentire a un maggior numero di utenti di accedere a risorse quali *database* e *file system*. Fondamentalmente, esistono due tipi di ***benchmark*** generici:

- **sintetici/artificiali**: non riflettono l'uso effettivo del ***server*** in un ambiente produttivo. Questi ***benchmark*** applicano inoltre un carico costante e artificiale a uno specifico componente del ***server***, in modo da valutarne le prestazioni massime. Spesso i risultati di questi ***benchmark*** non hanno un grande significato, se non quando vengono usati in modo relativo.
- **mondo/vita reale**: cercano invece di misurare le prestazioni offerte dal sistema sotto i carichi che potrebbero esistere realmente in un ambiente produttivo. Purtroppo, l'unico modo per compiere questo ***benchmark*** è quello di mettere il ***server*** in produzione e vedere che cosa succede - un lusso che ben pochi si possono permettere.



L'immagine illustra un benchmark (richieste soddisfatte al secondo) di richiesta di pagine ASP alterando il numero di processori di un sistema

I guasti del sistema possono dipendere dai *driver software* applicativi o a livello di sistema, da una configurazione errata, oppure dalla presenza di *hardware* difettoso. Quando si effettua il *burn-in*, spesso si possono rilevare dei problemi che altrimenti si manifesterebbero soltanto dopo un periodo di uso esteso, oppure quando il sistema si trova a dover gestire carichi elevati. Sul mercato sono disponibili molti ***benchmark***, decisamente troppi per elencarli tutti.

Alcuni sono di livello commerciale, altri sono *shareware*; alcuni sono particolarmente interessanti, mentre altri non hanno nessuna utilità pratica.

Burn-in di sistema e Benchmark

Attualmente, l'uso più comune dei ***benchmark*** è quello di misurare le prestazioni. I ***benchmark*** si possono tuttavia usare anche per il burn-in di un elemento *hardware*, oppure di una nuova **applicazione**. Nel corso di questa procedura, al ***server*** vengono imposti elevati livelli di sollecitazione per lunghi periodi di tempo, in modo da vedere se qualche parte del sistema si guasta.

I guasti del sistema possono dipendere dai *driver software* applicativi o a livello di sistema, da una configurazione errata, oppure dalla presenza di *hardware* difettoso. Quando si effettua il *burn-in*, spesso si possono rilevare dei problemi che altrimenti si manifesterebbero soltanto dopo un periodo di uso esteso, oppure quando il sistema si trova a dover gestire carichi elevati. Sul mercato sono disponibili molti ***benchmark***, decisamente troppi per elencarli tutti.

Transaction processing performance council. Chi usa grossi sistemi orientati alle transazioni ha probabilmente già sentito parlare dei ***benchmark*** *TPC* (*Transaction Processing Performance Council*). Questi ultimi sono diventati lo *standard* di fatto per la misurazione delle prestazioni dei sistemi basati sulle transazioni. I test *TPC* sono piuttosto complessi e difficili da configurare ed eseguire. Alcuni produttori, tra cui *Sybase* e *Compaq* hanno *team* dedicati all'esecuzione dei test *TPC* allo scopo di ottimizzare i prodotti per ottenere i migliori risultati. I risultati offerti da questi test vengono verificati da *TPC*: non esistono quindi risultati *TPC* non ufficiali.

Spec (*Standard Performance Evaluation Corporation*). È un ***benchmark*** sintetico che misura le capacità di elaborazione del ***server*** sugli interi e a virgola mobile. Questo ***benchmark*** viene usato spesso dai produttori di *CPU*, come *Intel* e AMD, per indicare le prestazioni relative dei microprocessori. Il ***benchmark*** SPEC è diffuso anche nel campo dell'informatica scientifica e tecnica. Esistono cinque tipologie principali di ***benchmark*** SPEC:

- SPEC CINT92: misura le prestazioni dell'elaborazione sugli interi.
- SPEC CFP92: misura le prestazioni dell'elaborazione a virgola mobile.
- SPEC CINT95: nuova versione del ***benchmark*** sull'elaborazione degli interi.
- SPEC CFP95: nuova versione del ***benchmark*** sull'elaborazione a virgola mobile.
- SPEC SDM: *Benchmark Software Development Multitasking*.

I fattori da prendere in considerazione nei benchmark

Quando ci si appresta a effettuare un ***benchmark*** è necessario prendere in considerazione i seguenti fattori.

- Verificare che ogni componente sia aggiornato.
- Mantenere una base uniforme.

- Rendere coerente il test.
- Ripetere il test.
- Tenere sotto controllo il ***server***, la rete e il ***benchmark***.
- Capire innanzitutto cosa fa il ***benchmark***.
- Controllare i risultati campione prima di iniziare.

I sistemi informatici sono complessi; anche il più piccolo ***server*** con processore singolo è formato da molti componenti, che devono lavorare all'unisono per assicurare l'operatività. Quando si esegue un ***benchmark***, bisogna essere certi che il sistema operativo, i *driver* delle applicazioni, il *firmware* specifico al ***server*** e i *driver* siano aggiornati. È possibile che il solo aggiornamento di un *driver* consenta di aumentare fino al 25 per cento le prestazioni del sistema.

Lo scopo del ***benchmark*** è quello di consentire un raffronto tra le prestazioni di due o più sistemi. A questo scopo, è necessario mantenere una base uniforme. Anche le più piccole variazioni nei sistemi possono distorcere i risultati. Per esempio, se si ha un sistema con quattro *hard disk* in una batteria RAID 0 e un altro sistema con sei dischi in una batteria RAID 0, il secondo godrà di un grosso vantaggio in lettura/scrittura, dal momento che usa più dischi per compiere la stessa quantità di lavoro.

Durante il ***benchmark***, la configurazione di ciascun ***server*** che viene sottoposto a test deve essere uniforme. Molti ***benchmark*** consentono una messa a punto secondo le proprie esigenze. Sebbene questa possibilità permetta di ottenere un'enorme flessibilità, esiste il pericolo di compiere un test incoerente. È quindi opportuno documentare tutte le modifiche che sono state apportate al ***benchmark*** e ripeterle per tutti i ***server*** sottoposti a test. La cosa migliore è compiere il test su ogni singolo ***server*** almeno per tre volte, osservando i dati statistici relativi alle prestazioni. Per esempio, se viene eseguito un test su un *database* caratterizzato da un elevato livello di *I/O*, è opportuno verificare che il sottosistema disco non rappresenti un collo di bottiglia.

Quando si esegue il ***benchmark***, bisogna tenere sotto controllo non soltanto il sistema operativo e le applicazioni eseguite sul ***server***, ma anche il *tool* di ***benchmark*** e i vari componenti del ***server***.

Dopo aver terminato ciascun test, è opportuno controllare i *log* degli errori del sistema operativo e dell'**applicazione**, in modo da verificare che non si sia verificato nessun problema. Capire che cosa fa il ***benchmark*** è la chiave per valutare con successo il ***server*** sottoposto al test.

Non è una buona idea quella di sottoporre a *benchmarking* un ***server*** multiprocessore con un solo *hard disk*, dal momento che non si potranno mai valutare le vere prestazioni del ***server***.

Dopo avere eseguito il ***benchmark*** sul ***server*** di test, bisogna scrutinare i risultati. Se viene usato un *tool* di *monitoring* del ***server*** in grado di registrare alcuni dati statistici a livello di sistema, come l'utilizzo della *CPU*, l'*I/O* su disco e la *cache* di memoria, controllare che il ***benchmark*** solleciti correttamente il sistema e che non esistano dei colli di bottiglia imprevisti. Al termine del ***benchmark*** è necessario valutare come si è comportato il sistema, usando i *tool* di *monitoring*. Bisogna essere certi che il ***server*** sia configurato appropriatamente allo scopo di ottenere i risultati che si stanno cercando. Diversamente si potrebbero trarre conclusioni errate.

I colli di bottiglia

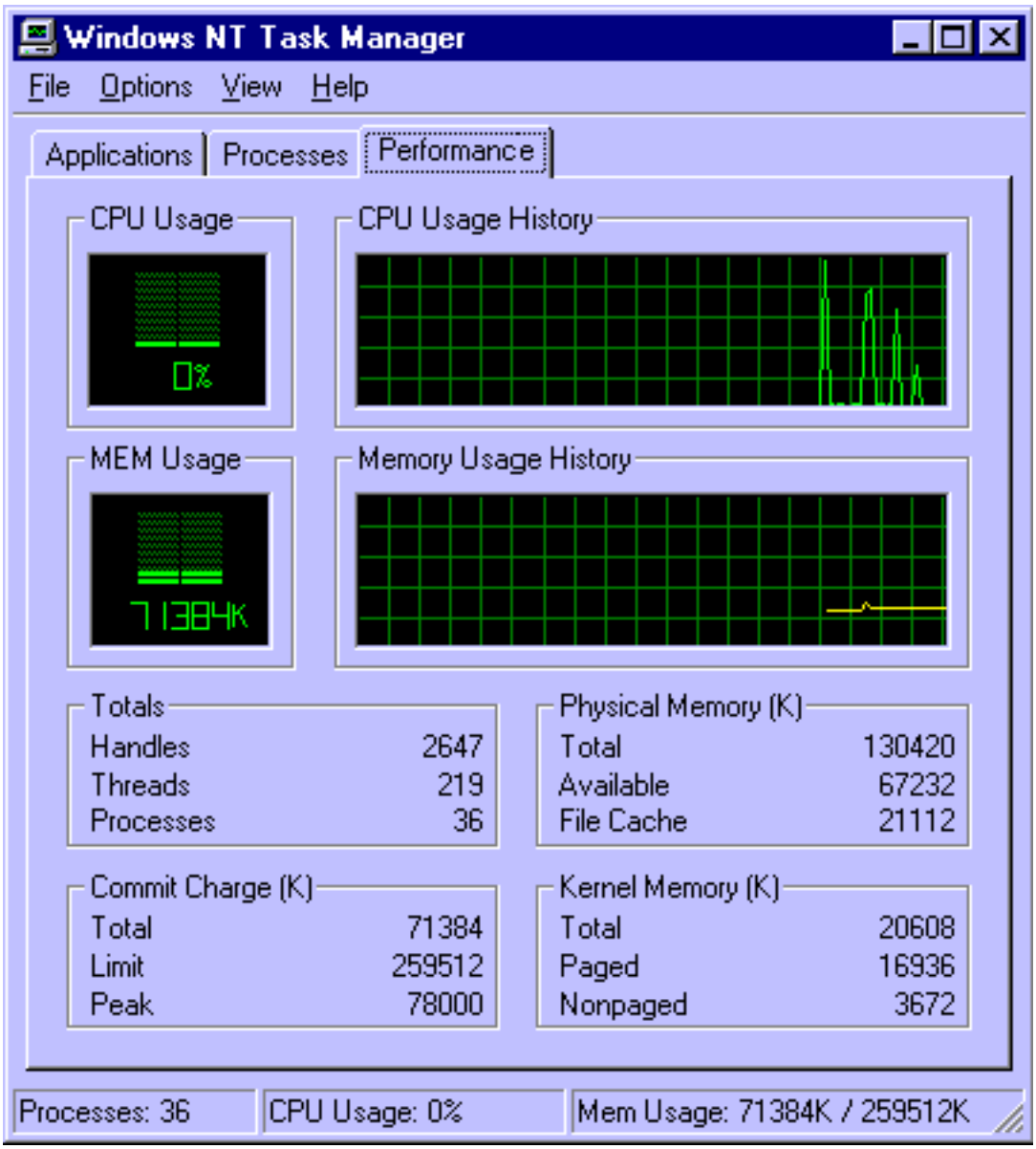
I colli di bottiglia del sistema possono esistere ovunque. Quando viene eliminato un collo di bottiglia, se ne manifesta subito un altro. Spesso il ***server*** è veloce soltanto quanto il collegamento o sub-componente più lento. Vi sono dei ***server*** equipaggiati molto bene, con processori multipli, centinaia di *Mbyte* di memoria e batterie di dischi ad alte prestazioni collegato alla rete attraverso un singolo segmento *Ethernet* da 10 Mbps. In casi come questo, spesso gli amministratori si domandano il perché le prestazioni per gli utenti sembrino così limitate se il ***server*** viene utilizzato soltanto al 5 per cento. Quando si cerca di identificare i colli di bottiglia, è opportuno esaminare i singoli componenti del sistema e il modo in cui questi componenti sono collegati.

La *CPU* e la memoria

Grazie ai nuovi processori, quasi sempre la *CPU* non costituisce più un collo di bottiglia. È tuttavia opportuno verificare attentamente il modello di *CPU* che viene utilizzato. Le dimensioni della *cache* di secondo livello possono avere effetti significativi sulle prestazioni. Se i *computer* vengono utilizzati come ***server*** di applicazioni, è opportuno scegliere una *cache* di secondo livello più capiente possibile.

Un altro importante componente del sistema è la memoria. Quando l'**applicazione** cerca i dati e questi ultimi non sono presenti nella *cache*, la *CPU* deve usare la memoria principale. Sebbene la memoria sia economica e molto veloce, la *cache* della *CPU* è ancora più veloce ma più costosa. La memoria virtuale consente di ingannare il sistema operativo, facendogli credere che sia disponibile più memoria di quanta ne esista effettivamente, ma questa memoria aggiuntiva viene archiviata sul disco. Quando il sistema operativo ha bisogno di dati che non sono presenti nella *cache* della *CPU* o nella memoria principale, legge il *file* della memoria virtuale su disco. Se il sistema operativo deve leggere in continuazione i contenuti del disco per ottenere i dati che gli servono, le prestazioni ne soffrono. Questa condizione viene chiamata **paginazione** e indica che è giunto il momento di aggiungere RAM al sistema.

Sono disponibili alcune linee guida generiche per stabilire quanta memoria dovrebbe essere installata su di un sistema. Una regola generale è quella di analizzare, in condizioni di carico il *task manager* ed osservare quanto la memoria usata (RAM + virtuale) sia lontana dalla RAM effettivamente installata. Anche il carico della *CPU* fornisce un primo sistema di analisi dello stato di salute del sistema.



Finestra di Windows che visualizza l'utilizzo del processore e della memoria

I fattori che possono determinare la quantità di memoria aggiuntiva che potrebbe rendersi necessaria sono diversi. Bisogna chiedersi quanti sono gli utenti che dovranno collegarsi, quanto spazio su disco occorre, se il ***server*** dovrà eseguire qualche **applicazione** (come un *database*) e quali sono le caratteristiche dell'uso del ***server***.

I dischi

I fattori che possono influenzare le prestazioni del disco sono molteplici. Le stesse caratteristiche dei dischi sono molto importanti. La velocità di rotazione è quella con cui ruotano i singoli piatti del disco. Una velocità di rotazione elevata è opportuna quando si usano applicazioni che richiedono l'uso di dati a flusso continuo, come quelli audio o video.

È opportuno tenere presente anche il numero di dischi che viene utilizzato. Se servono

8 *Gbyte* di spazio su disco, l'acquisto di un singolo *drive* di questa capacità non è la soluzione migliore in termini di prestazioni e di *fault tolerance*. Una soluzione migliore potrebbe essere quella di usare quattro dischi da 2 *Gbyte*; con un numero maggiore di dischi ci sono più piatti in rotazione ed è quindi possibile scrivere una maggiore quantità di dati allo stesso tempo. È importante anche il modo in cui vengono organizzati i vari dischi. Lo schema RAID 0 non offre la *fault tolerance*, ma permette di ottenere prestazioni molto valide. Lo schema RAID 1 offre la *fault tolerance* e buone prestazioni, ma è caratterizzato da un costo più elevato.

Il *controller* è un ulteriore componente del sottosistema a disco. L'implementazione più comune è costituita da un *controller* posto all'interno del ***server*** e collegato ai dischi tramite un *bus* SCSI. Il *controller* usa tipicamente una *cache on-board*, che aiuta ad aumentare le prestazioni mantenendo nella *cache* i dati usati più di recente, eliminando così la necessità di usare le unità a disco per recuperare i dati. Le dimensioni della *cache* possono variare da meno di 1 *Mbyte* a più di 64 *Mbyte*.

Strumenti di *monitoring*

Sul mercato sono disponibili molti strumenti per identificare i colli di bottiglia.

La procedura di identificazione dei colli di bottiglia è più un'arte che una scienza esatta. Tutti i sistemi operativi commerciali forniscono qualche serie di *tool*: NT offre il suo ***Performance Monitor***, mentre *NetWare* usa il *Monitor* NLM. Quasi tutti questi *tool* che accompagnano i sistemi operativi offrono funzionalità a livello di base. Per ottenere funzioni e informazioni più dettagliate, bisogna tuttavia rivolgersi ai *tool* di terze parti.

Negli ambienti *Unix*, molti amministratori usano i *tool* forniti con il sistema operativo centrale. *Utility* come ps e vmstat vengono usate piuttosto comunemente. Per *Sun Solaris* è disponibile una popolare *utility* chiamata iostat, che offre valide informazioni sulle prestazioni di *I/O*.

Tutti i *tool* dedicati alle prestazioni dovrebbero consentire di misurare le prestazioni del ***server*** con l'andare del tempo. Non è sufficiente prendere soltanto qualche campione, ma è necessario misurare le prestazioni entro un certo periodo di tempo, in modo da disporre di una base di misura.

Bibliografia

Introduzione

Libri

Zwicky, Cooper, Chapman*Building Internet Firewalls;* 2002*O'Reilly & Associates*

Ben Laurie, Peter Laurie*Apache La guida;* 1999Apogeo

Autori vari*Microsoft Site Server 3.0 Commerce Edition;* 2001Mondadori

Siti

Microsoft (IIS); http://www.microsoft.com/WINDOWS2000/iis/

Apache; http://www.apache.org

Firewall in WindowsXP;
http://www.microsoft.com/italy/windowsxp/home/using/howto/homenet/

Proxy servers; http://compnetworking.about.com/library

Glossario

Access List : Indica un elenco di regole di filtraggio del traffico dati, mediante cui è possibile abilitare l'accesso verso reti e applicazioni in maniera selettiva. Tipicamente una lista di accesso viene configurata su un *router* posto al confine fra reti, in modo da realizzare un punto di controllo del traffico di transito.

Access Provider : (Fornitore di accesso). Qualsiasi organizzazione che, essendo collegata direttamente a ***Internet***, vende ad altri la possibilità di accedervi.

ACL : (*Access Control List*). Abbreviazione per *Access Control List*, ovvero ***Access List***. Nella tecnologia delle reti *Microsoft Windows* rappresenta l'elenco delle regole di accesso ad una risorsa (esempio cartella del *file system*) e delle restrizioni attive su tale risorsa. Nell'ambito del *networking* il termine indica un filtro sul traffico che un *router* o un ***firewall*** effettua a scopo di protezione di una rete, di un'**applicazione**, di una macchina.

ActiveX : Una tecnologia che si prefigge gli stessi scopi di ***Java*** ma non ad architettura aperta (è di proprietà della *Microsoft* e permette la realizzazione di moduli di codice incorporabili tramite OLE).

Anonymous : Utente che non rivela la propria identità. Per convenzione viene designato con l'*account anonymous*.

Anonymous FTP : Una modalità di accesso ad un *computer* remoto tramite protocollo **FTP** che consente di accedere a *file* archiviati in siti pubblici. Specificando *anonymous* come *user* ID all'atto del *login*, non viene attivata la procedura di autenticazione dell'utente. La *password* da specificare, solitamente, è il proprio indirizzo *E-mail*.

Apache : Rappresenta un **HTTP** *server* multipiattaforma.

API : (*Application Program Interface*). L'API è un'interfaccia *software* utilizzata per la comunicazione fra programmi o per interfacciare livelli di **protocollo** diversi in un'architettura di comunicazione.

Applet : Un piccolo programma che può essere prelevato velocemente dalla rete e usato da qualsiasi *computer* dotato di un *browser* capace di eseguire codice ***Java***.

Applicazione : Un programma (*software*) che svolge determinate funzioni per l'utente finale. Esempi di applicazioni sono i *client* FTP, Telnet, *E-mail* e i *browser*.

ASP : Un ambiente di *scripting* per *Microsoft Internet Information Server*, in cui è possibile combinare **HTML**, **script** e componenti riusabili di **ActiveX**, per creare pagine *Web* dinamiche.

Benchmark : Test che intende mostrare la *performance* di un prodotto *hardware* o *software*.

CGI : (*Common Gateway Interface*). È un metodo con cui un *server* **HTTP** interagisce con *database*, documenti, e altri programmi inviando o ricevendo dati in formato **HTML** sul *browser client*. L'adozione di questo schema di elaborazione distribuita comporta l'esecuzione *server-side* di un programma (***CGI*** *script*) che effettua l'elaborazione e restituisce la risposta in **HTML** al ***client***. Molto spesso si può capire che viene usato un programma ***CGI*** se compare il termine *cgi-bin* nell'**URL** della pagina.

Client : Un programma usato per ottenere dati da un programma ***server*** residente su un altro *computer* situato da qualche parte nel mondo. Ogni programma ***client*** è progettato per colloquiare solo con uno o più particolari tipi di programmi ***server***, ed ogni ***server*** richiede un determinato tipo di ***client***.

Cookie : Un meccanismo che consente a programmi attivi sul ***server*** (per esempio, i programmi ***CGI***) di immagazzinare e recuperare dati sul lato ***client*** della connessione. Ciò significa che quando si accede di nuovo ad un certo **URL**, il ***server*** può riutilizzare i dati presenti direttamente sulla macchina ***client***. Qualche esempio: dati relativi al *login*, a registrazioni, ad acquisti in linea, eccetera. Poiché i ***cookie*** possono conservare informazioni utente sul *computer* di quest'ultimo, sono spesso usati per riconoscerlo e personalizzargli l'ambiente. Tipicamente, le informazioni memorizzate dai ***cookie*** scadono dopo un certo periodo di tempo.

DNS : (*Domain Name System*). Sistema/servizio per l'associazione e la traduzione di indirizzi numerici *IP* in nomi logici alfanumerici mnemonici. È basato su una struttura gerarchica ad albero, ad ogni ramo del quale corrisponde un dominio.

Firewall : Dispositivo comprendente componenti *hardware* e *software* preposto al filtraggio di pacchetti a scopo di protezione di una rete, di specifiche macchine, di specifiche applicazioni.

FTP : (*File Transfer Protocol*). Protocollo definito nella RFC 959. Rappresenta un modo comune per il trasferimento di *file* tra due *computer Internet*. Impiega il servizio di trasporto offerto da *TCP*.

GET : Istruzione data ad un ***server*** HTTP per richiedere una risorsa. Al contrario del metodo ***POST***, restituisce sempre una risorsa al ***client***.

HEAD : Istruzione data ad un ***server*** HTTP, simile a ***GET***, per recuperare gli attributi di una risorsa.

Host : Nell'architettura di rete *TCP/IP*, sinonimo di *end system*, cioé di sistema di elaborazione in una rete di *computer* origine/destinazione di dati applicativi. Il termine è utilizzato per distinguere un ***host*** da un **nodo** di commutazione (*router*).

HTML : (*Hyper Text Markup Language*). Linguaggio di realizzazione di ipertesti (interpretato), utilizzato per la realizzazione di pagine *Web* trasmesse mediante **protocollo** applicativo **HTTP**. Una pagina **HTML** può contenere testo, immagini, brani audio e sequenze video con vari formati e trasmessi come *file* dal ***server*** al ***client***.

HTTP : (*Hyper Text Transport Protocol*). Il trasferimento di *file* multimediali su ***Internet*** necessita del protocollo **HTTP** per il ***server*** e per il ***client***. Risulta il più diffuso **protocollo** applicativo per ***Internet***, al momento e si basa sul servizio fornito da *TCP*.

HTTPS : (*Secure* HTTP). Un'estensione del protocollo **HTTP** che permette di effettuare transazioni sicure, dati crittografati e autenticazione del mittente. Il **protocollo** enfatizza la massima flessibilità nella scelta della chiave da usarsi per la crittografia.

IIS : (*Internet Information Services*). Rappresenta un **HTTP *server*** multipiattaforma sviluppato da *Microsoft*.

Indirizzo : Stringa che identifica univocamente un'entità di rete.

Indirizzo *Internet* : Indirizzo a 32 bit assegnato alle interfacce degli ***host*** e dei *router* che utilizzano l'architettura di rete *TCP/IP*; lo si scrive come quattro numeri decimali separati da punti.

Internet : *internet* (con la i minuscola) indica una qualsiasi connessione tra due

reti. Con la I maiuscola: la più grande rete di calcolatori al mondo, basata sull'architettura di rete *TCP/IP*.

Java : Un linguaggio di programmazione *Object Oriented*, sviluppato da *Sun Microsystems* e disponibile già da diversi anni, specificatamente progettato per la scrittura di programmi che possono essere scaricati sul proprio *computer* dalla rete ed immediatamente eseguiti localmente. Utilizzando piccoli programmi ***Java*** (chiamati ***Applet***), le pagine *Web* possono includere animazioni, effettuare calcoli e quant'altro.

Javascript : Mentre un programma scritto in ***Java*** va sottoposto ad un **processo** di meta-compilazione per poter essere eseguito, ***Javascript*** è un linguaggio interpretato che può essere inserito direttamente nel codice **HTML** dei documenti *Web*.

LAN : (*Local Area Network*). Rete di calcolatori ad estensione locale/aziendale/dipartimentale, caratterizzata da mezzi trasmissivi condivisi, alta velocità trasmissiva, basso tasso di errore.

Linguaggio di *Scripting* : Un termine generico per qualunque linguaggio che è debolmente tipato o senza tipi, e non consente di utilizzare strutture dati complesse. Un programma in questo linguaggio è di solito interpretato.

Mailing list : Un sistema (solitamente automatizzato) che, ricevuto un messaggio *E-mail* da un utente, lo invia a tutti i componenti registrati di una lista. Così facendo è possibile partecipare a discussioni su vari argomenti.

MIME : (*Multipurpose Internet Mail Extension*). Estensione per il **protocollo** di posta elettronica ***Internet***, utilizzato per inviare insieme al messaggio, altri componenti non di testo (programmi eseguibili, immagini, documenti con formati binari, eccetera).

Name server : Un ***server*** sulla rete che ha il compito di tradurre i nomi delle macchine in indirizzi *IP*. Vedi anche: **DNS**.

Nodo : Ogni singolo calcolatore connesso in una rete.

Nome di dominio : Il nome di un *computer* che lo identifica univocamente su ***Internet***. I nomi di dominio hanno sempre 2 o più parti, separate da un punto, per esempio: oneweb.ibm.it. La parte più a sinistra è quella più specifica e quella a destra è quella più generale e costituisce il dominio vero e proprio.

PDU : (*Protocol Data Unit*). Unità informativa caratteristica di un **protocollo**.

POST : Istruzione data ad un ***server*** HTTP per richiedere di accettare una richiesta di un ***client***.

Processo : È la visione dinamica di un programma. Corrisponde all'insieme di azioni compiute da un programma in esecuzione su un sistema di elaborazione e all'insieme delle risorse che ad esso vengono allocate.

Protocollo : Insieme di regole definite per consentire la comunicazione di dati fra elaboratori.

Proxy : Componente *software* e/o *hardware* utilizzato per varie funzioni: per esempio per controllare da un punto unico l'accesso ad ***Internet*** e per analizzare i pacchetti *IP* che ***Internet*** accedono all'interno di una rete privata. Un **proxy** può avere anche le funzioni di traduzione di indirizzi *IP* privati in indirizzi *IP* pubblici e viceversa.

Request : Nel modello di riferimento OSI, primitiva di servizio attivata per richiedere la trasmissione di una **PDU**.

Response : Nel modello di riferimento OSI, primitiva di servizio dei protocolli che prevedono *acknowledge* attivata per indicare l'avvenuta ricezione sul **nodo** remoto di una **PDU** precedentemente trasmessa tramite una primitiva ***request***.

Script : Un programma scritto in un **linguaggio di *scripting***.

Server : Un *computer* o un programma che fornisce un determinato tipo di servizio ad un programma ***client*** in esecuzione su un *computer* remoto. Una stessa macchina può eseguire contemporaneamente più di un programma fungendo quindi da più ***server*** per molti ***client*** sulla rete.

Servlet : Un programma per ***server*** che garantisce funzionalità aggiuntive ai ***server*** abilitati ***Java***.

Socket : Nell'ambito dell'architettura *TCP/IP* indica un connettore logico, ossia un identificatore di una connessione logica fra un ***client*** ed un ***server Internet***. Un **socket** è rappresentato da una quadrupla di valori: **indirizzo** *IP* sorgente, *TCP/UDP port* sorgente, **indirizzo** *IP* destinatario, *TCP/UDP port* destinatario.

SSL : (*Secure Socket Layer*). È uno strato di *software* che si posiziona fra *TCP* ed una **applicazione** e consente di gestire un canale sicuro di comunicazione fra ***client*** e ***server***. La cifratura dei dati avviene mediante algoritmi crittografici asimmetrici a chiave pubblica.

Tag : Marcatore, racchiuso tra parentesi angolari, che costituisce l'elemento caratterizzante l'**HTML** (per esempio: <h1>, </H1>, <IMG SRC="stella.gif">).

URL : (*Universal Resource Locator*). Indica l'indirizzo logico di una pagina su ***Internet***, specificando il nome del ***server*** ed il percorso nel suo *file system* fino alla pagina specifica. Il formato per una **URL** è: [tipo applicazione]://[macchina]/[nome del *file*].

Autori

Hanno realizzato il materiale di questo modulo:

Prof. Cosimo Laneve

Professore Straordinario di Informatica presso l'Università di Bologna, dove insegna Linguaggi di Programmazione e Qualità del *Software*. Ha ricevuto il Dottorato di Ricerca in Informatica dall'Università di Pisa ed è stato *Research Associate* presso l'INRIA di *Sophia Antipolis* in Francia. Attualmente è coordinatore di progetti nazionali e internazionali che riguardano i fondamenti teorici e l'implementazione di linguaggi di programmazione distribuiti, di verifica statica di programmi e di teoria dei tipi. Relativamente a queste tematiche, ha pubblicato su numerose riviste e conferenze internazionali.