

1

Apache e Internet

Questo capitolo è un'introduzione ad Apache, HTTP (*Hypertext Transfer Protocol*) e alle problematiche delle reti. È destinato a coloro che non hanno mai utilizzato Apache e i server di rete in generale. La maggior parte dei concetti descritti nel capitolo è nota a chiunque abbia qualche esperienza di amministrazione di sistemi oppure abbia una buona conoscenza di argomenti connessi a Internet. Questi lettori potrebbero anche passare direttamente al Capitolo 2.

Apache necessita di un hardware adeguato, quindi in questo capitolo discuteremo della scelta dell'hardware del server e dei criteri più importanti da tenere presenti in questa scelta. Sebbene sia piuttosto semplice installare manualmente Apache, daremo anche uno sguardo alle soluzioni di server dedicati basati su Apache per coloro che cercano soluzioni già pronte, compresa l'assistenza da parte dei produttori. Infine completeremo il capitolo con uno sguardo ad alcuni strumenti di configurazione grafica disponibili per le installazioni di Apache.

Apache: anatomia di un server di rete

In questo paragrafo introdurremo alcuni dei concetti fondamentali sottesi alla definizione di server di rete. Daremo uno sguardo al funzionamento di Apache a un livello di base e alle funzioni che esso svolge. Cercheremo anche di comprendere le ragioni del costante successo di Apache come server di rete per Internet.

Il codice sorgente di Apache

Apache è noto per essere il software per server Web più diffuso. Il vero segreto del successo di Apache è la sua natura di progetto open-source, con il codice sorgente accessibile liberamente e gratuitamente. Ciò ne permette l'elaborazione da parte di diverse persone e offre opportunità ineguagliabili a coloro che vogliono aggiungere delle funzionalità ai loro server Web. Infatti, alcuni dei moduli più importanti di Apache sono nati da progetti sviluppati esternamente. `mod_vhost_alias` e `mod_dav` ne sono due buoni esempi. L'utilizzo del codice sorgente di Apache non solo è possibile, ma viene attivamente incoraggiato. Tutte le distribuzioni binarie attualmente contengono una copia completa del codice sorgente pronta per essere elaborata. L'esame del codice sorgente è istruttivo e a volte consente di scoprire un bug, tramite le cosiddette open peer-review. Se un utente individua un bug in Apache, può segnalarlo e inviare un fix (correzione) alla comunità di Internet e al team di sviluppatori di Apache. Le attività di pe-

er-review tramite Internet si traducono in uno sviluppo più rapido del server e dei moduli di terze parti, oltre a rendere disponibili in un minor tempo dei fix che permettono di ovviare ai bug scoperti.

La licenza Apache

Come per la maggior parte dei codici sorgente disponibili su Internet, Apache è protetto da una licenza che ne permette la distribuzione. A differenza della maggior parte dei codici sorgente, tuttavia, Apache non è fornito con la GNU Public License (GPL) ma possiede una licenza propria, molto più permissiva della GPL, in quanto permette una gamma più ampia di applicazioni commerciali e contiene solo alcune clausole fondamentali.

In generale, se vogliamo utilizzare Apache per i nostri scopi non dobbiamo preoccuparci di niente. Se intendiamo distribuire, identificare con il suo marchio o vendere una versione di Apache o un prodotto che includa Apache come componente, entra in gioco la licenza. Tuttavia, questa è un'approssimazione che non può essere generalizzata; se avete dei dubbi, leggete la licenza. La licenza di Apache è molto breve ed è contenuta in una sola pagina. È compresa in tutte le distribuzioni di codice sorgente e binario ed è riprodotta per comodità nell'Appendice C.

Tenete presente che ci sono diversi prodotti di terze parti basati su Apache forniti con la loro licenza, quindi in tal caso la licenza di Apache non si applica.

Come ottenere assistenza per Apache

Apache non prevede un'assistenza diretta da parte di ASF (*Apache Software Foundation*), sebbene sia possibile segnalare bug e problemi dopo essere ricorsi inutilmente a tutte le altre fonti di informazione. Come per la maggior parte dei progetti open-source, la migliore fonte di assistenza è il supporto informativo e nel contempo informale della comunità on-line. Per molte applicazioni esso risulta sufficiente, poiché l'affidabilità di Apache è tale da limitare la necessità di assistenza in casi di emergenza.

In particolare, i server Apache non hanno bisogno di "emergency fix", come avviene di solito per altri server di rete basati su Windows. Il fatto che Apache sia il più popolare tra tutti i server di rete è un chiaro indice della sua versatilità (alcuni dati statistici sulla sua diffusione e popolarità sono disponibili presso: <http://www.netcraft.com/survey/>).

Tuttavia, se l'assistenza rappresenta un problema, vi sono alcune opzioni disponibili.

- ❑ Nella linea di prodotti WebSphere di IBM Apache viene utilizzato come componente del core su AIX, Linux, Solaris e Windows NT. IBM offre assistenza per le proprie versioni di Apache, denominate IBM HTTPD Server.
- ❑ Apple Computers ha integrato Apache nel server di MacOS X e nei sistemi operativi desktop di MacOS X come componente standard di sistema. Poiché MacOS X si basa su un derivato di BSD UNIX, Apache su MacOS X è in realtà quasi identico a una tipica installazione BSD o Linux.
- ❑ Esiste inoltre il server di rete HP v.2.0.0 basato su Apache su hp-ux 11.0 e 11i (PA-RISC), ancora non distribuito.

Inoltre, ci sono vari produttori di server di rete Apache basati su Linux (per esempio RedHat) che offrono assistenza per i loro prodotti, in particolare per Apache. Come per la maggior parte dei servizi di assistenza, la qualità cambia a seconda dei produttori. Fortunatamente, e soprattutto nel caso di Linux, è facile trovare on-line le opinioni degli utenti sull'affidabilità dei vari produttori.

Infine, alcuni ISP (*Internet Service Provider*) e integratori di sistemi forniscono assistenza per Apache e sono elencati sul sito Web di Apache (<http://www.apache.org/info/support.cgi>). Il numero di ISP che offrono i server basati su Apache è cresciuto considerevolmente negli ultimi anni; essi presentano una vasta gamma di soluzioni, comprendente server dedicati, collocazione, server virtuali e hosted account (account ospitati).

Come funziona Apache

Apache non funziona come un'applicazione utente quale un elaboratore di testi. Opera dietro le quinte, fornendo servizi per le altre applicazioni che comunicano con esso, come, per esempio, un browser Web.

Nella terminologia UNIX, le applicazioni che forniscono servizi invece di comunicare direttamente con gli utenti sono definite daemon. Apache funziona come daemon anche su Windows NT, in cui però il daemon viene chiamato servizio. Windows 95/98 e Windows ME non sono in grado di eseguire Apache come servizio; esso dev'essere eseguito tramite la riga di comando (il prompt di MS-DOS o il comando Esegui dal menu Start...), sebbene Apache non interagisca con l'utente nel corso dell'esecuzione.

Apache è destinato a operare su una rete; pertanto, le applicazioni che dialogano con Apache possono trovarsi su un computer diverso da quello su cui funziona Apache. Queste applicazioni sono generalmente note come **client**. Ovviamente, la definizione di rete è molto generica; essa può essere costituita da un'intranet locale o dall'intera Internet, a seconda dello scopo del server. Nel corso del presente capitolo parleremo in modo più dettagliato delle reti, di come funzionano e di come Apache le utilizza per dialogare con i client.

Il tipo più comune di client è un browser Web (quasi sempre, parlando di client, intendiamo i browser). Tuttavia, ci sono diversi client che non sono browser: i più importanti sono i web robot e i crawler che indicizzano i siti Web.

Il compito principale di un server Web consiste nel tradurre una richiesta in una risposta adeguata a seconda delle circostanze prevalenti in quel momento. Qualunque sia il client, quando inizia la comunicazione con Apache esso invia ad Apache una richiesta di accesso a una risorsa. In risposta, Apache fornisce tale risorsa, ovvero una risposta alternativa che spiega come mai la richiesta non ha potuto essere esaudita. In molti casi, la risorsa è una pagina Web in HTML che risiede su un disco locale, ma questo è il caso più semplice. Ci possono essere molte altre possibilità: un file di immagini, il risultato di uno script che genera un output HTML, un'applet Java e così via.

Apache utilizza il protocollo HTTP per dialogare con i client. Si tratta di un protocollo request-response, cioè che definisce le modalità con cui i client possono fare le richieste e le modalità di risposta dei server. L'eseguibile di Apache prende il nome dal protocollo e sui sistemi UNIX è generalmente eseguito con il nome `httpd`, che sta per *HTTP daemon*. Discuteremo in seguito degli elementi fondamentali dell'HTTP; i dettagli costituiscono più o meno il resto del libro.

Confronto tra Apache su UNIX e Apache su Windows

Apache è stato originariamente scritto per essere eseguito su server UNIX e oggi lo si ritrova comunemente su Solaris, Linux e FreeBSD su Internet. Successivamente Apache è stato trasportato su Windows 95 e NT e ha fatto significative incursioni nel territorio dominato dai server di Microsoft e Netscape (una conquista non da poco, considerato lo strapotere di queste società).

A causa delle sue origini legate a UNIX, Apache 1.3 non ha mai funzionato su Windows altrettanto bene che su UNIX, perciò i programmatori hanno colto l'occasione dell'uscita di Apache 2.0 per ridisegnare completamente il core del server Apache. Un importante cambiamento è costituito dall'estrazione della logica specifica della piattaforma con sua incorporazione in un modulo separato, noto come MPM (*Multi Processing Module*). Pertanto, le implementazioni di Apache rispetto a Windows ora si avvantaggiano di un MPM dedicato per le loro piattaforme e funzionano più rapidamente e in modo più affidabile.

Apache funziona diversamente sui sistemi UNIX rispetto a quelli Windows. Su UNIX, Apache funzionava tradizionalmente come un processo duplicato. Una volta avviato, Apache 1.3 crea (o duplica) diversi nuovi processi figli per gestire le richieste del server Web. Ogni nuovo processo creato in questo modo rappresenta una copia completa del processo originario Apache. Apache 2.0 fornisce tale comportamento nell'MPM `prefork`, messo a punto per assicurare la compatibilità con Apache 1.3.

Windows non ha niente che assomigli alla chiamata del sistema di forking fornita da UNIX, pertanto Apache è stato ampiamente riscritto per utilizzare i thread nativi di Windows. Teoricamente si tratta di una soluzione molto più efficiente e leggera, poiché i thread possono condividere le risorse (riducendo pertanto le esigenze di memoria). Inoltre, permette un passaggio molto più intelligente da un task all'altro da parte del sistema operativo. Tuttavia, con Apache 1.3 è stato utilizzato il livello di emulazione Windows POSIX (uno standard di compatibilità UNIX) per implementare i thread, per cui tale versione non ha mai funzionato bene quanto avrebbe dovuto in teoria. Apache 2.0 usa direttamente i thread nativi di Windows e quindi funziona molto meglio.

Apache 2.0 prevede il supporto dei thread riferiti alla piattaforma UNIX, sotto forma degli MPM worker e perchild, che forniscono diversi modelli di elaborazione a seconda delle nostre necessità. La nuova architettura, associata ai vantaggi della programmazione a thread, assicura migliori prestazioni e riduce inoltre le differenze tra Windows e UNIX, semplificando in tal modo il futuro lavoro di sviluppo su entrambe le piattaforme.

Apache è più stabile su Windows NT, 2000 e XP che su Windows 9x/ME, soprattutto a causa dell'implementazione più efficiente dei thread su queste piattaforme. Se intendiamo eseguire Apache con Windows contando su un minimo di affidabilità, è meglio scegliere Windows NT, poiché permette anche ad Apache di funzionare come servizio di sistema.

Tuttavia, se l'affidabilità e la sicurezza sono di vitale importanza, c'è un'unica soluzione: utilizzare un server UNIX, sia per il buon funzionamento di Apache sia per quello del server in generale. Inoltre, le nuove versioni di Apache raggiungono lo stadio di stabilità molto più velocemente su UNIX che su Windows. Per tali motivi, consigliamo di scegliere UNIX per usufruire al più presto dei miglioramenti nell'uso del server.

Apache è in grado di funzionare su molti sistemi operativi (nella maggior parte dei casi, a partire da una distribuzione binaria installata), in particolare su OS/2, 680x0 e Macintosh basati su Power PC (sia le versioni precedenti sia quelle successive a MacOS X), BeOS e Netware. MacOS X si distingue soprattutto per essere quasi completamente "privo di segni particolari", poiché si tratta solo di una variante di UNIX. La dotazione di Apache 2.0 comprende MPM per UNIX, Windows, OS/2, BeOS e Netware; sono inoltre in via di sviluppo altri MPM. Per evitare inutili digressioni, non ci dilungheremo su questo; i dettagli sono presenti sul sito Web dell'ASF (<http://www.apache.org/>).

Discuteremo un po' più a lungo della scelta del sistema operativo nel resto di questo capitolo, nel paragrafo "La scelta dell'hardware del server".

File di configurazione e direttive

Apache viene impostato tramite file di configurazione, in cui è possibile inserire direttive per controllarne il comportamento. Apache supporta un numero impressionante di direttive e ogni modulo che viene aggiunto al server ne fornisce ancora altre.

L'approccio utilizzato da Apache per la configurazione lo rende estremamente versatile e assicura all'amministratore il completo controllo su tutte le funzionalità e le caratteristiche di sicurezza del server. Si tratta di uno dei pregi maggiori di Apache rispetto ai suoi concorrenti, che non offrono lo stesso grado di flessibilità ed estensibilità. Questo pregio ha una contropartita, la curva di apprendimento lievemente più ripida di Apache; ma gli sforzi sono ben ripagati con il controllo quasi completo su ogni aspetto delle operazioni del server Web.

Il rovescio della medaglia della versatilità di Apache è costituito dal fatto che, a differenza di altre offerte commerciali, non esiste al momento una soluzione completa per configurare Apache con un editor GUI, pertanto la configurazione dev'essere modificata manualmente. Detto questo, si deve aggiungere che sono stati compiuti alcuni tentativi credibili per creare uno strumento di configurazione valido, al fine di sfruttare tutta la potenza e la flessibilità di Apache. Uno di questi potrebbe rivelarsi adatto ai nostri bisogni; la sua scelta sarà condizionata dalle nostre esigenze. Per maggiori informazioni, consultate il paragrafo "Strumenti grafici di configurazione" nel Capitolo 2.

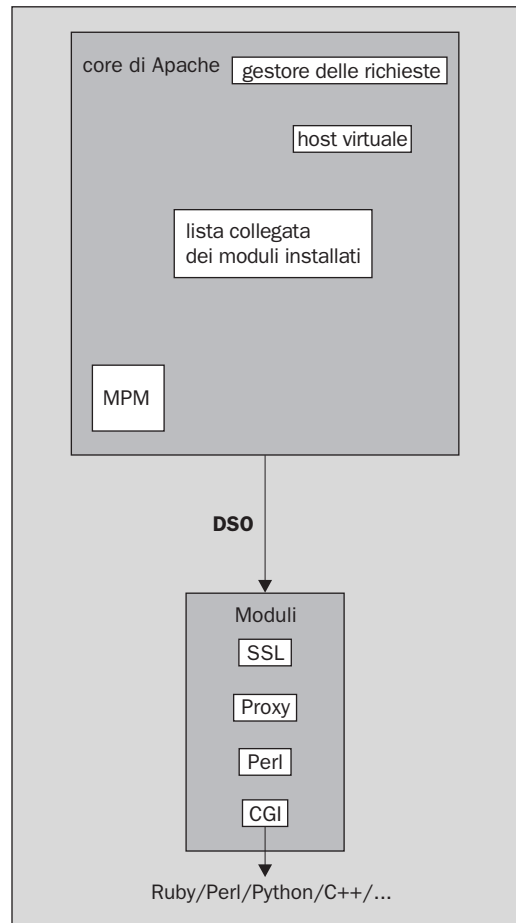
Questo libro si occupa, per la maggior parte, delle direttive di configurazione di Apache. Presentiamo le più importanti nel Capitolo 4 e le più avanzate nel resto del libro, introdotte nel contesto delle funzionalità che forniscono. Tuttavia, dedicheremo del tempo anche all'esame di

questioni più marginali: il Capitolo 3 riguarda la costruzione di Apache dal sorgente, durante la quale si possono utilizzare alcune opzioni di configurazione aggiuntive non disponibili in altre fasi; nel Capitolo 10, invece, si analizza la questione della sicurezza del server Web dal punto di vista dell'intero server, non solo di Apache. Si tratta di un argomento che esula dalla semplice configurazione di Apache.

Moduli

Uno dei più importanti punti di forza di Apache risiede nella sua struttura modulare. Nell'eseguibile principale di Apache c'è soltanto un insieme di funzionalità di core; tutto il resto è fornito da moduli che possono essere integrati in Apache o caricati dinamicamente mentre esso è in funzione. In Apache 2.0 questo concetto viene ulteriormente ampliato, portando le funzionalità specifiche delle piattaforme negli MPM e suddividendo i moduli monolitici come `mod_proxy` e `mod_cache` in moduli secondari core e di implementazione specifica, come mostrato nella Figura 1.1. Questo permette di scegliere precisamente le sole funzionalità desiderate e fornisce anche un'architettura estensibile per nuovi tipi di proxy e cache.

Figura 1.1



Di conseguenza, l'amministratore del server di rete può scegliere quali moduli includere ed escludere mentre costruisce Apache dal sorgente, rimuovendo le funzionalità superflue, il che rende il server più piccolo, meno avido di memoria e meno soggetto a errori di configurazione, in definitiva più sicuro. In alternativa, i moduli normalmente assenti da Apache possono essere aggiunti e attivati per fornire maggiori funzionalità.

Apache consente inoltre di aggiungere i moduli dinamicamente, in modo da non dover essere costretti a ricostruire Apache quando decidiamo, in un secondo tempo, di aggiungere nuove funzionalità. Per aggiungere un nuovo modulo, basta semplicemente installarlo e riavviare il server di Apache. Per supportare i moduli caricati dinamicamente, Apache necessita di un po' più di memoria e il server si avvia più lentamente, perché deve caricare i moduli dal disco. Si tratta di uno svantaggio minimo ma che può essere significativo se sono richieste prestazioni elevate.

È disponibile una vasta gamma di moduli di terze parti per Apache. Alcuni di essi, come `mod_fastcgi`, forniscono caratteristiche aggiuntive specifiche, molto utili a chi voglia rendere Apache più potente. Con `mod_fastcgi`, Apache diventa in grado di memorizzare nella cache gli script CGI in modo tale da renderli più reattivi nei confronti degli utenti e meno esigenti in relazione alle risorse del sistema.

Altri moduli assicurano maggiore potenza e flessibilità. `mod_perl` integra un interprete Perl completo in Apache, permettendogli di utilizzare tutta la gamma di software disponibili per Perl. Alcuni moduli, precedentemente di terze parti, sono stati aggiunti alla distribuzione Apache 2.0, in particolare `mod_dav`. Tuttavia, il new entry più importante è la base del supporto SSL in Apache: `mod_ssl`. Attualmente integrato nella distribuzione di Apache, questo modulo elimina una serie di inconvenienti (tra cui l'esigenza di correggere con una patch il codice sorgente di Apache) che in passato gli amministratori del server di rete dovevano subire.

La sua flessibilità, associata alla stabilità e alle buone prestazioni, nonché alla disponibilità del codice sorgente, rende Apache la scelta più frequente tra i software per i server Web su Internet.

L'Hypertext Transfer Protocol

L'HTTP è il protocollo sottostante utilizzato da tutti i server Web e dai client. Mentre l'HTML definisce il modo in cui le pagine Web sono descritte, l'HTTP si occupa del modo in cui i client eseguono le richieste di informazioni e di come i server rispondono.

Di solito, l'HTTP opera a un livello sotterraneo, ma una comprensione del suo funzionamento a livello elementare può essere utile all'amministratore del server Web per diagnosticare i problemi e affrontare le questioni relative alla sicurezza. Molte delle funzionalità di Apache sono collegate all'HTTP; pertanto la comprensione di questo protocollo sarà utile anche per conoscere meglio tali funzionalità.

Il protocollo HTTP/1.1 è definito nei dettagli in RFC 2616 (<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>).

L'HTTP è un protocollo stateless di gestione richieste/risposte; ciò vuol dire che il dialogo tra un client Web (che può essere un browser o meno) e il server consiste in una richiesta da parte del client, a cui il server invia una risposta, eseguendo tutte le elaborazioni intermedie necessarie.

Richieste e risposte dell'HTTP

La prima riga di una richiesta dell'HTTP è costituita da un metodo che descrive ciò che il client vuole fare, un URI (*Uniform Resource Identifier*), che indica la risorsa da recuperare o manipolare, e una versione del protocollo HTTP. A questo segue un numero opzionale di intestazioni che possono modificare la richiesta in vari modi, per esempio per subordinarla al soddisfacimento di determinati criteri oppure per specificare un nome host (richiesto in HTTP/1.1). Ad avvenuta ricezione della richiesta e di eventuali intestazioni accompagnatorie, il server stabilisce come intervenire e risponde alla richiesta in modo adeguato. Una richiesta tipica di un documento HTML potrebbe essere:

```
GET /index.html HTTP/1.1
Host: www.alpha-complex.com
```

*Usando il comando `telnet` o un'utility di connessione a riga di comando di tipo simile, potete connettervi a un server attivo e digitarla manualmente per vedere la richiesta e la risposta direttamente (per esempio: `telnet localhost 80`). Premete il tasto **Invio** o **Return** due volte per*

inviare la richiesta dopo aver digitato entrambe le righe. Per maggiori dettagli sull'utilizzo di telnet, consultate il Capitolo 2.

Le richieste che hanno avuto buon esito restituiscono un codice di status di 200 e le informazioni richieste precedute dalle intestazioni di risposta del server. Di seguito viene riportato un tipico insieme di intestazioni di risposta per un server Apache:

```
HTTP/1.1 200 OK
Date: Mon, 09 Aug 2002 06:37:56 GMT
Server: Apache/2.0.34 (UNIX)
Last-Modified: Mon, 02 Aug 2002 06:31:48 GMT
ETag: "d456-68-248fdd00"
Accept-Ranges: bytes
Content-Length: 104
Content-Type: text/html; charset=ISO-8859-1
```

La riga di status, contenente il tipo di protocollo e il codice relativo all'esito, appare per prima, seguita dalla data e da alcune informazioni sul server, a loro volta seguite dal resto delle intestazioni di risposta (che variano a seconda del server e della richiesta). Il più importante è il Content-Type, che dice al client che cosa fare della risposta, mentre il Content-Length fa sapere al client quanto è lungo il corpo della risposta. Le intestazioni Date, ETag e Last-Modified sono usate nel caching.

Se si verifica un errore, nella riga di stato apparirà un codice di errore e il motivo dell'errore:

```
HTTP/1.1 404 Not Found
```

È anche possibile che in certi casi, per esempio in caso di redirectione, il server restituisca altri codici.

I metodi dell'HTTP

Il metodo comunica al server il tipo di richiesta fatta e può essere uno di quelli riportati di seguito. Gli esempi sono abbreviati al minimo necessario per illustrare la natura della richiesta e della risposta. Un autentico server Apache invierà probabilmente molte più intestazioni di quelle illustrate nell'esempio seguente:

Metodo	Funzione	Esempio
GET	Acquisizione di un'intestazione e una risorsa dal server. Una riga vuota separa l'intestazione dalla risorsa.	Richiesta: GET /index.html HTTP/1.0 Risposta: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 17:02:08 GMT Server: Apache/2.0.33 (UNIX) Content-Length: 1776 Content-Type: text/html; charset=ISO-8859-1 Connection: close <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <html> ... </html>

(Segue)

Metodo	Funzione	Esempio
HEAD	Restituzione dell'intestazione che verrebbe acquisita con un GET, ma non della risorsa stessa. Si noti che la lunghezza del contenuto viene restituita anche se non c'è contenuto.	Richiesta: HEAD /index.html HTTP/1.0 Risposta: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 17:01:13 GMT Server: Apache/2.0.33 (UNIX) Content-Length: 1776 Content-Type: text/html; charset=ISO-8859-1 Connection: close
POST	Invio di informazioni al server. La risposta del server può contenere la conferma che l'informazione è stata ricevuta. Il server dev'essere configurato per rispondere in modo adeguato a un POST, per esempio con uno script CGI.	Richiesta: POST /cgi-bin/search.cgi HTTP/1.0 Content-Length: 46 query=alpha+complex&casesens=false&cmd=submit Risposta: HTTP/1.1 201 CREATED Date: Mon, 09 Aug 2002 17:02:20 GMT Server: Apache/2.0.33 (UNIX) Content-Type: text/html; charset=ISO-8859-1 Connection: close <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <HTML> ... </HTML>

Nell'HTTP/1.1 sono supportati anche i seguenti comandi:

Metodo	Funzione	Esempio
OPTIONS	Restituzione della lista dei metodi consentiti dal server. Ciò è molto importante per i server WebDAV che supportano metodi addizionali definiti in RFC 2518.	Richiesta: OPTIONS * HTTP/1.1 Host: www.alphacomplex.com Risposta: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 16:54:55 GMT Server: Apache/2.0.33 (UNIX) Allow: GET, HEAD, POST, OPTIONS, TRACE Content-Length: 0 Content-Type: text/plain; charset=ISO-8859-1
TRACE	Monitoraggio di una richiesta per sapere quello che vede realmente il server. L'obiettivo consiste nel vedere come la richiesta appare dopo essere passata attraverso un qualsiasi proxy intermedio.	Richiesta: TRACE * HTTP/1.1 Host: www.alphacomplex.com Risposta: HTTP/1.1 200 OK Date: Mon, 09 Aug 2002 17:09:18 GMT Server: Apache/2.0.33 (UNIX)

Metodo	Funzione	Esempio
	Può anche essere diretta a un proxy intermedio tramite l'intestazione Max-Request per scoprire informazioni sui server intermedi. Per maggiori informazioni su TRACE consultare RFC 2616.	<pre>Content-Type: message/http; charset=ISO-8859-1 TRACE * HTTP/1.1 Host: www.alphacomplex.com</pre>
DELETE	Cancellazione di una risorsa sul server. In generale il server non permette il metodo DELETE, perciò il tentativo di utilizzarlo può produrre una risposta come quella fornita nell'esempio. L'eccezione è costituita dai server WebDAV, che implementano la funzione DELETE.	Richiesta: <pre>DELETE /document.html HTTP/1.1 Host: www.alphacomplex.com</pre> Risposta: <pre>HTTP/1.1 405 Method Not Allowed Date: Mon, 09 Aug 2002 17:24:37 GMT Server: Apache/2.0.33 (UNIX) DAV/2 Allow: GET, HEAD, OPTIONS, TRACE Content-Type: text/html; charset=ISO-8859-1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <HTML><HEAD> <TITLE>405 Method Not Allowed</TITLE> </HEAD><BODY> <H1>Method Not Allowed</H1> The requested method DELETE is not allowed for the URL /document.html.<P> </BODY></HTML></pre>
PUT	Creazione o modifica di un file sul server. In generale, il server non consente il metodo PUT, poiché la maggior parte delle sue applicazioni può essere effettuata anche con il metodo POST. PUT implica una relazione diretta tra l'URI nella richiesta PUT e lo stesso URI in un successivo GET, ma esso non è sottinteso da POST. Anche in questo caso, il metodo può essere implementato dai server WebDAV.	Richiesta: <pre>PUT /newfile.txt HTTP/1.1 Host: www.alphacomplex.com Content-Type: text/plain Content-Length: 63 This is the contents of a file we want to create on the server</pre> Risposta: <pre>HTTP/1.1 201 CREATED Date: Mon, 09 Aug 2002 17:30:12 GMT Server: Apache/2.0.33 (UNIX) DAV/2 Content-Type: text/html; charset=ISO-8859-1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <HTML> ... </HTML></pre>
CONNECT	Un metodo che permette ai proxy di passare a una modalità tunneling per i protocolli come l'SSL.	Per maggiori dettagli, consultare la direttiva AllowCONNECT nel Capitolo 8.

URI

Un URI è una stringa di testo che identifica una risorsa, per nome, posizione o con qualsiasi altro formato comprensibile al server. Gli URI sono definiti in RFC 2396.

L'URI è solitamente un convenzionale URL (*Uniform Resource Locator*) così come viene compreso da un browser; nella forma più semplice, è costituito da una /. Tutti gli URI validi sul server possono essere specificati qui, per esempio:

```
/
/index.html
/centralcontrol/bugreport.htm:80
http://www.alpha-complex/images/ultraviolet/photos/outside.jpg
```

Se il metodo non richiede di accedere a una risorsa specifica, allora può essere utilizzato lo speciale URI *; l'esempio OPTIONS a fianco si riferisce a un'applicazione di *. Si noti che in questi casi non è sbagliato utilizzare un URI valido: è solo superfluo.

Protocollo HTTP

La versione del protocollo è una delle seguenti:

- ☐ HTTP/0.9
- ☐ HTTP/1.0
- ☐ HTTP/1.1

In pratica, non viene mai inviato un HTTP/0.9, perché l'argomento del protocollo è stato introdotto solo a partire dall'HTTP/1.0, per distinguere le richieste 1.0 da quelle 0.9. Si presume che si tratti di HTTP/0.9 se il client non invia un protocollo, ma solo GET e POST funzioneranno in questo modo poiché gli altri metodi sono stati introdotti solo a partire dall'HTTP versione 1.0.

Intestazioni dell'HTTP

Le intestazioni dell'HTTP (note anche come campi di intestazione HTTP) possono accompagnare i messaggi HTTP in transito tra client e server, in entrambe le direzioni. In pratica può essere inviata qualsiasi intestazione, se entrambe le estremità della connessione concordano sul suo significato, ma il protocollo HTTP definisce soltanto un sottoinsieme specifico delle intestazioni.

Le intestazioni HTTP riconosciute sono divise in tre gruppi.

- ☐ **Le intestazioni di richiesta** sono inviate dai client al server per aggiungere informazioni o modificare la natura della richiesta. L'intestazione `Accept-Language`, per esempio, informa il server dei linguaggi accettati dal client, utilizzabili da Apache a fini di negoziazione dei contenuti.
- ☐ **Il server invia intestazioni di risposta** ai client in **risposta** alle richieste. Le intestazioni standard normalmente inviate da Apache comprendono `Date` e `Connection`.
- ☐ **Le intestazioni di entità** possono essere inviate in entrambe le direzioni, aggiungendo informazioni descrittive (definite anche "metainformazioni") sul corpo di un messaggio HTTP. Le richieste HTTP possono utilizzare le intestazioni di entità solo per i metodi che permettono un corpo, cioè PUT e POST. Le richieste con un corpo sono obbligate a inviare un'intestazione `Content-Length` per informare il server della dimensione del corpo. I server invece possono inviare un'intestazione `Transfer-Encoding` ma devono altrimenti inviare un'intestazione `Content-Length`.

In aggiunta alle intestazioni di contenuto, che comprendono anche `Content-Language`, `Content-Encoding` e la ben nota `Content-Type`, vi sono altre due utili intestazioni di entità: `Expires` e `Last-Modified`. Con `Expires` si comunica ai browser e

ai proxy la durata della validità di un documento, mentre `Last-Modified` permette a un client di determinare se un documento memorizzato nella cache è aggiornato. Per illustrare l'utilità di tale funzione, immaginiamo un proxy con un documento memorizzato nella cache, per il quale giunge una richiesta. Il proxy invia prima al server una richiesta `HEAD` e cerca un'intestazione `Last-Modified` nella **risposta**. Se ne trova una che non sia più recente del documento nella cache, il proxy non si preoccupa di richiedere il documento al server ma invia invece la versione della cache. Per maggiori dettagli, vedere `mod_expires` nel Capitolo 4.

Nell'Appendice H è contenuta la lista completa delle intestazioni di HTTP riconosciute.

Il networking e il protocollo TCP/IP

Sebbene un computer possa essere isolato, è in genere molto più utile collegarlo a una rete. Perché un server Web sia accessibile, dev'essere collegato al mondo esterno.

Per collegare in rete due o più computer tra loro, è necessario un mezzo di comunicazione. In un ufficio tale mezzo è di solito Ethernet, che comprende una scheda di rete installata in ogni computer da inserire nella rete e i relativi cavi di connessione. Tuttavia, l'hardware da solo non è sufficiente. Sebbene sia possibile inviare dati tramite un semplice collegamento tra le porte seriali, i computer che sono in rete con molti altri computer necessitano di un protocollo più avanzato per definire le modalità di trasmissione, consegna, ricezione e riconoscimento dei dati.

Il protocollo TCP/IP (*Transport Communication Protocol/Internet Protocol*) è uno dei tanti protocolli utilizzati per la comunicazione tra i computer in rete, ed è il protocollo più usato in Internet. Vi sono poi altri protocolli, per esempio Token Ring (che non funziona su Ethernet) e SPX/IPX (che invece funziona). Entrambi sono di solito presenti in intranet aziendali.

Definizioni

In realtà, il TCP/IP non è costituito da un protocollo, ma da due, l'uno sovrapposto all'altro. Al livello inferiore, l'IP si occupa dell'instradamento dei dati tra mittente e destinatario, dividendo i dati in pacchetti e allegando un indirizzo di origine e uno di destinazione a ogni pacchetto, come nella spedizione per posta ordinaria, con i pacchetti contenuti in buste.

Attualmente sono disponibili due versioni di IP. La meno recente, e più comune, è l'IPv4 (*IP version 4*). Si tratta del protocollo con cui opera ancora la maggior parte di Internet; esso comincia tuttavia a mostrare i segni del tempo. Il suo successore è **IPv6**, che estende l'intervallo di indirizzi da 32 a 128 bit, aggiungendo inoltre il supporto dell'IP mobile per la telefonia cellulare e la determinazione della qualità del servizio, nonché l'autenticazione opzionale e la crittografia delle connessioni di rete. Questa parte del protocollo è abbastanza ampia da avere indotto gli sviluppatori di standard a pubblicarla in una specifica separata (nota come IPsec).

Il TCP si basa sull'IP per gestire i dettagli del trasferimento di una richiesta di dati da un punto all'altro. Inoltre il TCP fornisce i meccanismi per stabilire le connessioni, assicurando che i dati arrivino in ordine e gestendo la perdita di dati, gli errori e il recupero dei dati. Il TCP definisce un protocollo handshake per rilevare gli errori di rete e definire il proprio insieme di informazioni di envelope, compreso un numero di sequenza, che viene aggiunto al pacchetto di dati inviati dall'IP per assicurare che i dati arrivino nello stesso ordine in cui sono stati inviati.

Il TCP non è l'unico protocollo che utilizza l'IP. Della suite del protocollo TCP/IP fa parte anche UDP (*Unigram Data Protocol*). A differenza del TCP, che è un protocollo affidabile e orientato alla connessione, l'UDP è un protocollo senza connessioni e non garantito, utilizzato per la trasmissione di dati non critici e per i broadcast (le trasmissioni radiotelevisive), di solito per i messaggi che si adattano a entrare in un pacchetto. Poiché esso non controlla l'avvenuta trasmissione o la corretta sequenza, è utile in situazioni in cui il TCP sarebbe troppo ingombrante, come le trasmissioni radiotelevisive su Internet e i giochi multiplayer. Esso è anche la base delle reti peer-to-peer come Gnutella (<http://www.gnutella.com/>).

Il TCP/IP comprende anche l'ICMP (*Internet Control Message Protocol*), utilizzato dal software TCP/IP per comunicare i messaggi relativi al protocollo stesso, come, per esempio, il collegamento fallito a un determinato host. L'ICMP è destinato a essere utilizzato dal livello inferiore del protocollo TCP/IP e di rado viene destinato ad applicazioni di livello utente.

I protocolli TCP, UDP, ICMP e IP sono definiti nei seguenti RFC – UDP:768, IP:791, ICMP:792, TCP:793. Consultare l'Appendice A per una lista completa degli RFC e di altri documenti utili.

Pacchetti e incapsulamento

Abbiamo precedentemente menzionato il fatto che l'IP e il TCP funzionano entrambi con l'aggiunta di informazioni a pacchetti di dati trasmessi successivamente tra gli host. Per comprendere realmente il funzionamento del TCP/IP, è utile saperne un po' di più sul reale funzionamento dell'IP e del TCP.

Quando un'applicazione invia un blocco di dati, diciamo un file o una pagina in HTML, il TCP divide i dati in pacchetti per la trasmissione vera e propria. Lo standard Ethernet limita la grandezza massima dei pacchetti a 1500 byte e sulle reti più vecchie l'hardware potrebbe limitare la grandezza dei pacchetti a 576 byte. Quando stabilisce la connessione, il TCP/IP determina la grandezza dei pacchetti. Anche se la rete locale può gestire pacchetti da 1500 byte, potrebbe non essere così per la rete intermedia o di destinazione. A meno che in una fase intermedia non sia possibile suddividere i pacchetti, tutta la comunicazione dovrà essere uniformata al livello minimo di dimensione dei pacchetti.

Una volta stabilite le dimensioni dei pacchetti, il TCP incapsula ogni blocco di dati destinati a un pacchetto con un'intestazione TCP che contiene un numero di sequenza, le porte di origine e destinazione e un checksum per il rilevamento degli errori. Questa intestazione è analoga all'indirizzo su una busta, in cui la lettera è rappresentata dal pacchetto di dati.

L'IP aggiunge quindi la sua intestazione al pacchetto TCP, in cui registra gli indirizzi IP di origine e destinazione, in modo che gli stadi intermedi sappiano come instradare il pacchetto. Aggiunge anche un tipo di protocollo per identificare il pacchetto come TCP, UDP, ICMP o qualche altro protocollo e un altro checksum. Se stiamo usando IPv6, allora il pacchetto potrà essere firmato per autenticare il mittente e crittografato per la trasmissione.

Inoltre, se il pacchetto dev'essere inviato su una rete Ethernet, Ethernet aggiunge un'altra intestazione che contiene gli indirizzi di origine e destinazione Ethernet, un codice del tipo e un ulteriore checksum.

Questo avviene perché mentre l'IP registra gli indirizzi IP degli host mittenti e ricevuti nell'intestazione, Ethernet usa gli indirizzi Ethernet delle interfacce di rete per ogni fase del viaggio effettuato dal pacchetto. Ogni protocollo ha un raggio di azione minore rispetto a quello che viene incapsulato, per cui descrive salti sempre più brevi nel percorso dall'origine alla destinazione.

Sia l'IP sia il TCP aggiungono 20 byte di informazioni a un pacchetto di dati, che nel complesso deve rientrare nei limiti dei 1500 byte imposti da Ethernet. Pertanto, le dimensioni massime dei dati che possono essere contenuti in un pacchetto TCP/IP sono di soli 1460 byte. Ovviamente, se l'IP non funziona su Ethernet ma su una connessione seriale, non è necessario limitare il pacchetto a 1500 byte. Altri protocolli, naturalmente, possono imporre proprie limitazioni.

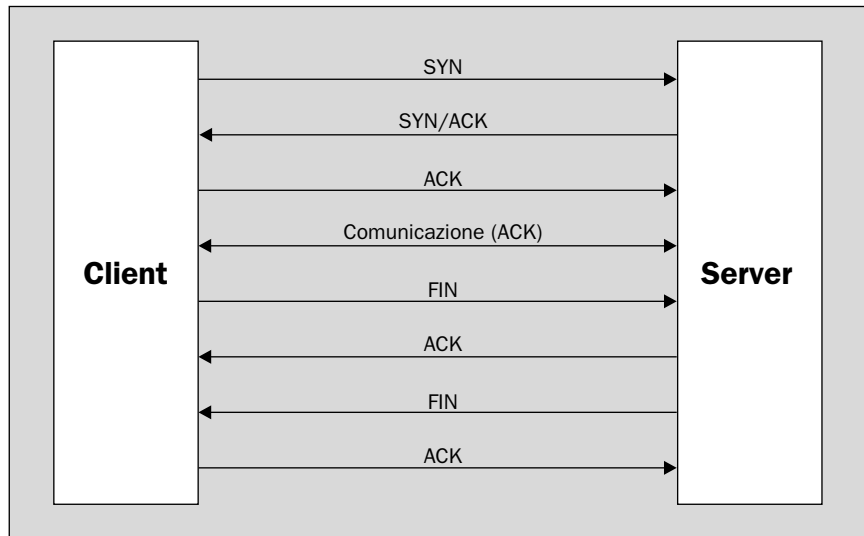
ACK, NAK e altri messaggi

La maggior parte delle trasmissioni TCP è composta da pacchetti di dati, come descritto in precedenza. Tuttavia, l'IP non fa alcun tentativo per assicurare che il pacchetto arrivi a destinazione, pertanto il TCP richiede che la destinazione invii un messaggio **Acknowledged** (ACK) per avvertire l'host mittente che il messaggio è arrivato. Gli ACK, pertanto, sono comuni quasi quanto i messaggi di dati e in una rete ideale il loro numero coincide con quello dei messaggi di

dati. Se la ricezione del pacchetto non va a buon fine, il TCP richiede che il destinatario invii invece un messaggio **Not Acknowledged** (NAK).

In aggiunta ai dati, agli ACK e ai NAK, il TCP definisce anche i SYN (sincronizzazione), per stabilire le connessioni, e i FIN per terminarle. Il client richiede una connessione inviando un messaggio SYN al server, che instaura o nega la connessione inviando rispettivamente un ACK o un NAK. Se una delle due parti della connessione vuole interrompere la comunicazione, invia all'altra un messaggio FIN per indicare tale intenzione. La Figura 1.2 illustra questo procedimento.

Figura 1.2



Ci sono quindi tre eventualità che l'host mittente può prevedere di dover affrontare.

- ☐ Quando l'host di destinazione riceve un pacchetto e se il pacchetto era quello che attendeva o una nuova connessione, invia un ACK.
- ☐ Se d'altro canto il checksum del pacchetto non corrisponde o il numero di sequenza del pacchetto è errato, il destinatario invia un NAK per informare l'host che deve inviare nuovamente il pacchetto.
- ☐ Se il destinatario non invia alcun messaggio, il TCP conclude che il pacchetto o la risposta sono probabilmente andati persi e li invia nuovamente.

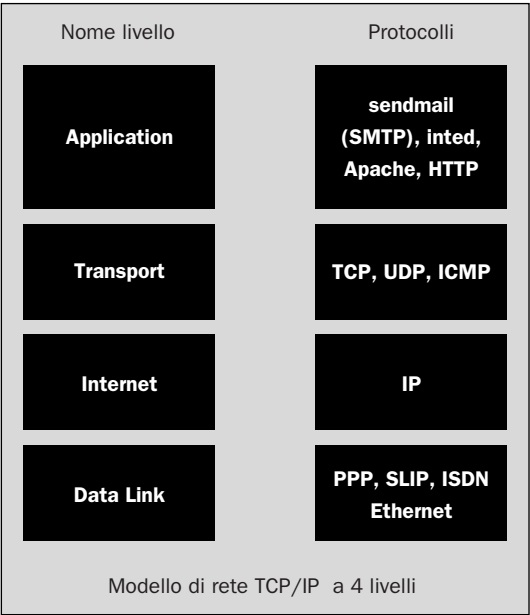
Vari tipi di attacco DOS (*Denial-Of-Service*) sfruttano taluni aspetti del protocollo TCP/IP per tentare di tenere occupati inutilmente i server. Uno di tali attacchi, il *SYN flood*, consiste nell'invio a un server di vari pacchetti SYN, mentre l'accettazione delle connessioni richieste non viene mai riconosciuta dal client. Di conseguenza, lo studio del funzionamento dei pacchetti TCP è tutt'altro che un soggetto accademico. Le azioni da intraprendere per difendersi da tali attacchi costituiscono l'argomento del Capitolo 10.

Il modello di rete TCP/IP

Il TCP e l'IP costituiscono due livelli in una gerarchia di protocolli che si estende dall'applicazione in cima fino all'hardware in fondo. Il modello di rete TCP/IP è una versione semplificata del modello OSI di networking a sette strati, cui somiglia, ma fino a un certo punto. Sebbene il modello OSI venga spesso menzionato e paragonato al TCP/IP nella documentazione sulle reti, il paragone è pressoché inutile, poiché non esiste niente che sia completamente compatibile con OSI. La comprensione del TCP/IP in sé è molto più importante.

Il TCP/IP è una gerarchia a quattro livelli, costruita sulle fondamenta dell'hardware e al di sotto dell'applicazione, con il TCP e l'IP in posizione intermedia. Il seguente diagramma semplificato (Figura 1.3) ne illustra la struttura.

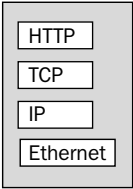
Figura 1.3



Il livello Data Link qui è mostrato come un livello singolo, sebbene in pratica contenga spesso più livelli. Tuttavia, il vantaggio del protocollo TCP/IP è che non dobbiamo occuparcene. Per esempio, in una comunicazione tipica tra un server di rete e il client, i livelli potrebbero essere disposti come segue.

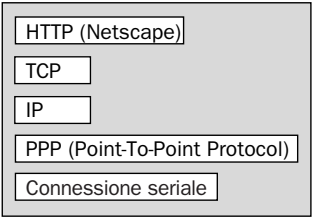
Al server, connesso a una rete Ethernet (Figura 1.4).

Figura 1.4



Al cliente, un utente su un account di rete dial-up (Figura 1.5).

Figura 1.5



In tal caso è stato utilizzato un protocollo PPP aggiuntivo per permettere all'IP di funzionare essendo sovrapposto al protocollo seriale di base utilizzato tra i modem, trasformando il livello inferiore data link in due livelli.

Se l'utente richiede una pagina Web tramite il proprio server, il browser genera la richiesta utilizzando il protocollo HTTP. Essa viene poi trasmessa tramite una connessione stabilita dal TCP, utilizzando l'IP per instradare il pacchetto contenente la richiesta a una gateway attraverso una connessione seriale, utilizzando il protocollo PPP.

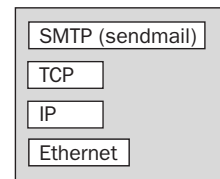
L'IP può instradare il pacchetto attraverso vari server intermedi. Le informazioni di indirizzo contenute nel pacchetto comunicano a ogni server intermedio dove esso dovrà essere indirizzato.

Nel server, l'interfaccia di rete vede un pacchetto il cui indirizzo IP lo identifica come destinato al server. Il server lo estrae dalla rete e lo invia al TCP, che identifica la richiesta come richiesta di connessione e la riconosce. Poco dopo, sulla rete transita un pacchetto di dati inviato nuovamente al TCP, che lo identifica come appartenente alla connessione appena stabilita. Esso riconosce il pacchetto di dati, rimuove le informazioni della busta e presenta l'acclusa richiesta di HTTP ad Apache.

Apache elabora la richiesta e invia una **risposta** al client, ripercorrendo la strada verso il basso nella gerarchia e riattraversando Internet in direzione del client.

Se invece cercassimo di gestire un sistema di posta sul server di posta elettronica UNIX, i livelli del protocollo sarebbero configurati come mostrato nella Figura 1.6.

Figura 1.6



Come possiamo vedere, l'unica differenza è costituita dal protocollo del livello superiore e dall'applicazione che usiamo; il TCP/IP gestisce per noi qualsiasi altro aspetto della transazione.

Protocolli non IP

Oltre all'IP, ci sono diversi altri protocolli che funzionano direttamente su Ethernet e non utilizzano il protocollo IP. Per esempio, l'ARP (*Address Resolution Protocol*) è utilizzato sulle reti Ethernet per dedurre l'indirizzo Ethernet di un'interfaccia di rete dal suo indirizzo IP. I protocolli rivali come SPX/IPX funzionano anche su Ethernet senza coinvolgere l'IP. La struttura di Ethernet permette a tutti questi protocolli di coesistere senza problemi.

Pochissimi di questi protocolli sono utilizzati su Internet, poiché la maggior parte di essi non è in grado di compiere il viaggio dal punto di partenza al punto di arrivo con più hop (salti), come avviene invece con il protocollo IP. Pertanto, i protocolli che hanno bisogno di tale funzione, come il TCP o l'UDP, sono basati su di esso invece che essere indipendenti.

Indirizzi IP e classi di reti

Ogni host in una rete TCP/IP necessita di un indirizzo IP unico, assegnato dagli amministratori di rete. Oltre a ciò, se l'host deve comunicare in Internet, deve avere un indirizzo IP unico in tutta Internet.

Gli indirizzi IPv4 sono numeri a 32 bit, scritti di solito a gruppi di quattro byte, o ottetti, con un valore compreso tra 0 e 255, separati da punti, come, per esempio, 192.168.20.181.

Gli indirizzi IPv6 sono numeri a 128 bit, rappresentati da blocchi esadecimali separati dai due punti, come, per esempio, fe80::910:a4ff:aefe:9a8. Si noterà che non ci sono cifre sufficienti a formare un indirizzo da 128 bit. Questo avviene perché un certo numero di zeri è stato compresso nello spazio occupato dai due punti, per cui gli zeri non vengono elencati esplicitamente. Tale numero è solo parzialmente sotto il nostro controllo; parte di esso deriva da un in-

dirizzo Ethernet dell'interfaccia di rete. Questo permette l'allocazione automatica degli indirizzi IPv6 e del networking IP mobile, uno degli obiettivi principali del progetto IPv6. L'IPv6 verrà trattato successivamente più nei dettagli, in questo stesso capitolo.

L'intervallo totale degli indirizzi IP è diviso in regioni entro le quali risiedono classi di rete differenti. Il resto di Internet considera gli indirizzi IP contenuti in una classe di reti come facenti parte della stessa rete e prevede di utilizzare un punto di contatto, la cosiddetta **gateway**, per indirizzare i pacchetti agli host di quella rete.

Oltre a ciò, alcuni indirizzi IP (il primo, costituito solo da zeri, e l'ultimo, costituito interamente da 255) di ogni classe sono considerati speciali, pertanto non esistono tanti indirizzi per host quanti ne immaginiamo. Tra poco daremo uno sguardo a questi indirizzi speciali.

L'intervallo di indirizzi IPv4, che è tuttora lo schema di attribuzione degli indirizzi utilizzato in Internet, è diviso nominalmente in regioni di reti di classe A, B e C per fini di allocazione.

- ❑ Reti di classe A: ne esistono molto poche e occupano l'intervallo di indirizzi il cui primo numero è compreso tra 1 e 126. Solo il primo numero è fisso e il numero totale di possibili host in una rete di classe A è 16.581.375.
- ❑ Reti di classe B: occupano l'intervallo compreso tra 128 e 191. Il primo e il secondo numero sono fissi e danno origine a un totale di 16.382 possibili reti di classe B, in ognuna delle quali sono possibili 65.560 host.
- ❑ Reti di classe C: sono le più piccole, occupano l'intervallo compreso tra 192 e 223. I primi tre numeri sono fissi e rendono disponibili oltre due milioni di reti di classe C, ma ognuna di esse può possedere solo 254 host.
- ❑ L'intervallo compreso tra 224 e 254 è riservato nella specifica TCP/IP.

Lo spazio di indirizzi IPv6 è diviso analogamente, ma in un intervallo più ampio: 6 ottetti (48 bit) sono fissi, mentre i rimanenti 10 (80 bit) sono assegnati alla rete locale.

Indirizzi IP speciali

Alcuni indirizzi IP sono considerati in modo speciale dalle reti TCP/IP. Entro una classe di rete, un indirizzo di zeri è utilizzato per connotare un indirizzo di origine anonimo, se l'host non sa di quale indirizzo IP si tratti; comunque, è un'evenienza rara. Un indirizzo di 255 cifre è un indirizzo di broadcast per la rete (tutti gli host sulla rete possono ricevere un broadcast). La maschera di rete non è necessariamente un indirizzo: essa definisce gli indirizzi di un intervallo di indirizzi IP che sono considerati come direttamente collegati tra loro (situati cioè sullo stesso segmento di rete). Gli indirizzi che differiscono per qualcosa di più della semplice maschera di rete sono situati su reti diverse e devono utilizzare gateway e router per comunicare.

A seconda della classe di rete, il numero di zeri o di 255 varia, come illustrato da questi tre esempi:

Classe	Anonimo	Broadcast	Maschera di rete
A	16.0.0.0	16.255.255.255	255.0.0.0
B	181.18.0.0	181.18.255.255	255.255.0.0
C	192.168.32.0	192.168.32.255	255.255.255.0

Poiché i broadcast sono privi di connessione (dal momento che l'host di origine invia i dati a ogni host in grado di riceverli), essi vengono eseguiti utilizzando il protocollo UDP. L'IPv6 funziona in modo diverso dall'IPv4 a tale riguardo; esso non supporta il broadcasting e utilizza invece il **multicasting**. Per semplicità, lasceremo da parte questo argomento e passeremo a trattare l'IPv4.

Ci sono anche alcuni intervalli di indirizzi IP che vengono trattati in modo particolare dall'hardware di networking come i router: gli indirizzi compresi in questi intervalli sono considerati privati e i pacchetti destinati a essi non vengono mai inviati fuori dalla rete loca-

le da parte dei router. Per tale ragione, questi indirizzi sono una buona scelta per controllare le reti o per le intranet che non saranno mai collegate direttamente a Internet. La lista completa è illustrata nella tabella seguente.

Classe	Reti private
A	10.0.0.0
B	172.16.0.0 to 172.31.0.0
C	192.168.0.0 to 192.168.255.0

Un altro speciale indirizzo IP è quello di loopback, **127.0.0.1**, che si riferisce all'host locale (spesso chiamato, in modo abbastanza appropriato, *localhost*). Lo si utilizza per accedere ai server che funzionano sul computer locale.

Gli altri indirizzi della rete che iniziano per 127 sono utilizzati dai server di posta per identificare gli open relay e altri mittenti di posta indesiderata. I servizi MAPS, ORDB, ORBZ e Spews utilizzano tutti server delle query dei DNS (*Domain Name System*), che restituiscono un indirizzo 127 quando l'indirizzo IP di origine è sulla "lista nera". L'operazione funziona perché l'indirizzo non è legittimo e quindi consente efficacemente al server DNS di fare una query yes-or-no. Si tratta di un utilizzo non standard ma efficace degli standard di attribuzione degli indirizzi TCP/IP.

Maschera di rete e instradamento

Gli indirizzi IP sono composti da due metà: l'indirizzo di rete e quello dell'host locale. Le classi di rete A, B, e C corrispondono a reti con un numero esatto di ottetti, ma possiamo usare una maschera di rete (*netmask*) per dividere la rete e l'indirizzo locale in un dato punto, utilizzando l'aritmetica binaria.

La maschera di rete è un numero che somiglia a un indirizzo IP ma non lo è. Essa viene invece sottoposta a un AND logico con un indirizzo IP per ottenere l'indirizzo di rete (**181.18.0.0** nell'esempio precedente per la classe B). Se i due indirizzi IP mappano sullo stesso indirizzo di rete dopo essere stati sottoposti al comando AND con la maschera di rete, allora si trovano sulla stessa rete; altrimenti, sono su due reti differenti. Le maschere di rete IPv6 non sono diverse dalle loro controparti IPv4, solo più lunghe e meno interessanti.

La maschera di rete è un attributo fondamentale dell'interfaccia di rete, tanto quanto l'indirizzo IP. Utilizzandola, il server può stabilire se l'indirizzo IP di destinazione è locale e quindi può essere contattato direttamente, oppure dev'essere raggiunto indirettamente tramite un router intermedio. Prendiamo, per esempio, tre host con i seguenti indirizzi IP:

Indirizzo IP	Host
192.168.1.1	Host a
192.168.1.2	Host b
192.168.2.1	Host c

Se definiamo una maschera di rete di **255.255.255.0** per le interfacce di rete su ogni host, si presumerà che i primi due host, host a e host b, siano sulla stessa rete. Se host a invia un pacchetto TCP/IP, cercherà di inviarlo direttamente a host b. Tuttavia, se host a cerca di inviare un pacchetto a host c, non potrà inviarlo direttamente poiché la maschera di rete stabilisce che **192.168.1** e **192.168.2** sono reti diverse; pertanto esso invierà il pacchetto a una gateway. Ogni host è configurato nell'indirizzo IP di almeno un gateway per poter inviare i pacchetti che non può consegnare autonomamente.

Se tuttavia definiamo una maschera di rete di **255.255.0.0**, tutti e tre gli host saranno considerati presenti sulla stessa rete. In tal caso, host a sarà in grado di effettuare un invio diretto a host c, posto ovviamente che essi siano realmente connessi alla stessa rete fisica.

L'IP ha il compito di garantire che un pacchetto indirizzato a un host particolare arrivi proprio a esso. Dividendo lo spazio dell'indirizzo in reti logiche, il compito di trovare un host diviene molto più semplice: infatti, invece di dover conoscere tutti gli host su Internet, per l'host è necessario solo conoscere una lista di gateway e scegliere quella che rappresenta il passo logico successivo sull'itinerario. L'identità del passo successivo viene poi fornita al protocollo sottostante (per esempio Ethernet) in modo che il pacchetto vi venga inoltrato. Nel caso di Ethernet, l'identità è rappresentata dall'indirizzo Ethernet della gateway sulla rete locale. Il gateway esegue la stessa procedura utilizzando la sua lista e così via, finché il pacchetto raggiunge il gateway finale e la propria destinazione.

Per vedere come funziona in pratica una maschera di rete, consideriamo l'esempio dell'output di ifconfig fornito successivamente nel capitolo. Esso mostra due indirizzi locali e un'interfaccia esterna Ethernet partizionata da una maschera di rete allo scopo di costringerle su reti separate.

Ricerca di servizi, porte ben note

Se un client contatta un server, ciò avviene solitamente perché vuole utilizzare un particolare servizio, per esempio l'e-mail o l'FTP. Per differenziare i due servizi, il TCP definisce il concetto di porte, permettendo a una singola interfaccia di rete di fornire vari servizi differenti. Se un client richiede a un server di connettersi alla rete, esso specifica non solo l'indirizzo IP del server che vuole contattare, come richiesto dal protocollo IP, ma anche un numero di porta.

Di default, i server HTTP come Apache servono la porta 80, che è il numero standard di porta del protocollo HTTP. Se arriva una richiesta di connessione alla porta 80, il sistema operativo sa che Apache sorveglia quella porta e indirizza le comunicazioni a essa. Ogni servizio di rete standard e ogni protocollo ha una porta associata a cui il client può collegarsi per quel servizio, sia esso HTTP, FTP, telnet o altro.

La lista standard dei numeri di porta è definita in UNIX in un file denominato `/etc/services`, che elenca tutti i numeri di porta allocati. Il file corrispondente sotto Windows è definito `Services` ed è situato nella directory di installazione di Windows `C:\WINNT\system32\drivers\etc\`. Infatti il sistema operativo e i vari daemon responsabili della fornitura dei servizi sanno già quali porte usano; `/etc/services` è destinato all'utilizzo di altre applicazioni, in modo che possano riferirsi a un servizio per nome e non tramite il numero. `/etc/services` specifica quale protocollo, TCP o UDP, viene usato dal servizio; molti servizi gestiscono sia le connessioni TCP sia quelle UDP. Di seguito viene fornita una breve lista di alcuni dei più comuni numeri di porta, estratti da un tipico file `/etc/services`:

ftp	21/tcp		# File Transfer Protocol
finger	79/tcp		# Finger Daemon
www	80/tcp	http	# WorldWideWeb HTTP
www	80/udp		# HyperText Transfer Protocol
pop-2	109/tcp	postoffice	# Post Office Protocol
pop-2	109/udp		# Version 2
pop-3	110/tcp		# Post Office Protocol
pop-3	110/udp		# Version 3
nntp	119/tcp	readnews untp	# USENET News Transfer Protocol
ntp	123/tcp		# Network Time Protocol
ntp	123/udp		#
imap2	143/tcp	imap	# Interactive Mail Access
imap2	143/udp	imap	# Protocol V2
snmp	161/udp		# Simple Net Management Protocol
imap3	220/tcp		# Interactive Mail Access
imap3	220/udp		# Protocol V3
https	443/tcp		# Secure HTTP
https	443/udp		# Secure HTTP
uucp	540/tcp	uucpd	# UNIX to UNIX Copy

In questa lista sono particolarmente interessanti la porta HTTP al numero 80 e la porta HTTPS al 443; da notare che entrambe le connessioni UDP e TCP sono accettate su queste

porte. La loro gestione durante l'utilizzo su una determinata porta dipende dal relativo programma. Il fatto che un servizio sia elencato non implica necessariamente che il server risponderà. In realtà, ci sono una serie di buone ragioni per non rispondere a determinati servizi: `telnet`, `ftp`, `snmp`, `pop-3` e `finger` non hanno niente a che vedere con la funzione di servire le pagine Web e potrebbero essere utilizzati per indebolire la sicurezza del server.

Sui sistemi UNIX, i numeri di porta inferiori a 1024 sono riservati ai servizi del sistema e non sono utilizzabili dai programmi eseguiti da utenti non privilegiati. Affinché Apache funzioni sulla porta 80, la porta standard per l'HTTP, dev'essere avviato dalla root o dal sistema operativo all'avvio del sistema. Gli utenti non privilegiati possono ancora eseguire un server Apache, nella misura in cui configurano Apache affinché funzioni con un numero di porta pari o superiore a 1024. Windows non offre tali condizioni di sicurezza.

inetd, il super server di networking

Non tutti i servizi forniti da un host sono gestiti da un daemon costantemente in funzione. Poiché ciò comporterebbe un grande spreco di risorse del sistema, UNIX esegue molti dei suoi servizi tramite **Internet Daemon** (`inetd`), un super server che ascolta varie porte differenti e avvia un programma che gestisce le connessioni man mano che le riceve.

L'FTP, uno di questi servizi, è solitamente eseguito sulla porta 21. A differenza di Apache, che normalmente funziona in modo indipendente e compare in forma di vari processi `httpd`, non esiste alcun processo `ftpd` operante in condizioni normali. Tuttavia `inetd` controlla la porta 21 e quando riceve una richiesta di connessione TCP avvia una copia di `ftpd` per gestire la connessione. Una volta avviata, `ftpd` negozia quindi la sua connessione privata con il client, consentendo a `inetd` di ritornare alla propria funzione di ascolto. Una volta terminata la comunicazione, nel caso dell'FTP, il file viene trasferito o abbandonato e il daemon termina.

Apache 1.3 ha una direttiva di configurazione, `ServerType`, che gli permette di funzionare come servizio indipendente o di essere richiamato da `inetd` come l'FTP. In questa configurazione non ci sono processi `httpd` in corso finché `inetd` non riceve una richiesta di connessione per la porta 80 (o qualsiasi altra porta configurata da `inetd` per avviare Apache). `inetd` quindi esegue `httpd` e gli consegna la connessione in arrivo, permettendo ad Apache di gestire la richiesta. Poiché per ogni connessione individuale del client viene avviata una richiesta separata di Apache, che dura solo il tempo necessario a soddisfarla, si tratta di un modo assolutamente inefficiente di eseguire Apache, ragion per cui quasi tutte le configurazioni di Apache sono indipendenti. Di conseguenza, in Apache 2.0 questa opzione è stata completamente cancellata.

`inetd` non è tuttavia privo di problemi. In veste di daemon centrale che coordina molti servizi di rete di minore importanza, esso rappresenta una delle maggiori fonti di violazione della sicurezza di rete. Il daemon in sé non è poco sicuro ma implementa servizi come `telnet`, che sono invece insicuri. Il risultato è che molti amministratori di server di rete scelgono di disattivarlo interamente, visto che nessuno dei servizi che gestisce è necessario per un server di rete. Le più recenti distribuzioni UNIX contengono un daemon migliorato, denominato `xinetd`, che incorpora misure di sicurezza aggiuntive, ma nella maggior parte dei casi non ci sono ragioni che costringano ad attivarlo. Per maggiori informazioni al riguardo, consultate il Capitolo 10.

Il futuro: l'IPv6

L'attuale protocollo IP, l'IPv4, utilizza 4 numeri a otto bit per formare gli indirizzi IP, permettendo un numero di indirizzi possibili pari a 2^{32} . Anche tenendo conto dell'esistenza di indirizzi anonimi e broadcast, ce ne sono in teoria abbastanza da fornirne uno a quasi ogni persona sul pianeta, sicuramente a chiunque abbia un computer. Purtroppo, a causa della divisione di tali indirizzi in classi di rete A, B, e C, gli indirizzi IP rischiano di esaurirsi.

La soluzione è costituita dall'IPv6, la versione 6 del protocollo IP, che fornisce indirizzi a 128 bit invece che a 32. Mentre gli indirizzi IPv4 sono scritti normalmente come quattro numeri decimali separati da punti, gli indirizzi IPv6 sono costituiti da otto numeri esadecimali a quattro cifre, separati da due punti. Gli zeri iniziali possono essere omessi e sostituiti da un doppio punto

e virgola per brevità, per cui un indirizzo IPv6 può avere la forma `fe80::910:a4ff:aefe:9a8`, che costituisce l'abbreviazione di `fe80:0910:0000:0000:a4ff:aefe:09a8`. Questo permette un numero vertiginoso di indirizzi IP possibili, ovvero 2^{128} .

L'IPv6 introduce anche il supporto per molte altre importanti caratteristiche. Una di esse è l'informazione sulla qualità del servizio, che permette di dare una priorità ai dati in rete. Questo consente ai server di gestire il traffico HTTP con priorità maggiore rispetto, per esempio, alla posta elettronica. Un altro esempio è costituito dall'autenticazione e dalla crittografia, fornita da IPsec, la specifica per la sicurezza integrata nel protocollo IPv6.

L'IPsec sostituisce l'SSL, ma può svolgere molte più funzioni, tra cui l'autenticazione e la consegna garantita di pacchetti individuali di informazioni. Esso rappresenta la base delle moderne VPN (Virtual Private Network) e merita di essere preso in considerazione da società che desiderano disporre di una soluzione sicura con cui estendere le loro intranet private a uffici remoti e computer mobili.

Il supporto dell'IPv6 è ora disponibile per la maggior parte delle piattaforme; su Linux e BSD esiste da più tempo, mentre le piattaforme commerciali si sono adeguate solo di recente. Apache 2.0 supporta attualmente gli indirizzi IPv6 in tutte le direttive che riguardano la rete, soprattutto `Listen`, `VirtualHost`, `allow` e `deny`. Tuttavia la reale implementazione delle reti IPv6 sta avvenendo in modo molto lento, nonostante i vantaggi che offre.

Comunque, l'adozione reale dell'IPv6 raggiungerà la massa critica solo se ci saranno server sufficienti a supportarla. Pertanto, riflettiamo sulla possibilità di aggiungere l'IPv6 alla configurazione di Apache, e se abbiamo un server presso un provider, incoraggiamo il provider ad aggiungere anche il supporto per l'IPv6. Se il provider non è ancora in grado di supportare l'IPv6, facciamo pressioni su di esso affinché lo faccia o passiamo a un provider che se ne occupa.

L'IPv6 è essenzialmente una rete separata che funziona accanto all'IPv4 e su una propria rete. La rete principale che supporta l'IPv6 durante l'installazione e l'esecuzione è nota come dorsale IPv6 (**6bone**) e i suoi punti di accesso sono disponibili nella maggior parte dei Paesi. Ci sono tre modi per ottenere un indirizzo IPv6 ed entrare a far parte della rete IPv6.

- ❑ Ottenere un indirizzo 6bone allocato tramite un provider che già fa parte del 6bone. Questi indirizzi sono assegnati essenzialmente da 6bone.
- ❑ Ottenere un indirizzo IPv6 di produzione da un ISP con un identificatore di rete IPv6 top-level di produzione. Questi indirizzi vengono assegnati dal RIR (*Regional Internet Registry*).
- ❑ Utilizzare un tunnel di collegamento tra IPv6 e IPv4 per collegare un indirizzo IPv4 locale a un indirizzo esterno IPv6. Gli indirizzi in questo intervallo iniziano per 2002, seguito dall'indirizzo IPv4 del router sulla rete locale. I bit rimanenti formano la porzione locale dell'indirizzo IPv6 e sono allocati dal provider.

Per maggiori informazioni su 6bone e IPv6, nonché per le istruzioni dettagliate su come stabilirsi in una rete IPv6, consultate <http://www.6bone.net/> e in particolare le pagine che spiegano come entrare a far parte di 6bone.

Strumenti di networking

L'amministrazione di una rete è un processo complicato, tanto da non poter essere discusso in questa sede; alcuni aspetti dell'amministrazione vengono trattati, dal punto di vista delle prestazioni e della sicurezza, rispettivamente nei Capitoli 8 e 10. Tuttavia, ci sono alcune utility valide che possono essere talvolta impiegate da un amministratore di server di rete nella diagnosi e risoluzione dei problemi. UNIX è generalmente meglio equipaggiato per questo tipo di analisi rispetto alla maggior parte degli altri sistemi operativi, essendosi evoluto di pari passo con Internet ed essendo diventato il sistema operativo predominante per l'implementazione dei sistemi Internet.

Gli strumenti di networking di UNIX contengono varie utility che possono essere impiegate dall'amministratore di rete. Gli strumenti che ora descriveremo consentono di trovare facilmente le informazioni utili.

ifconfig

`ifconfig` è un'utility standard su tutti i sistemi UNIX e riguarda la configurazione delle interfacce di rete (`if` è l'abbreviazione di interfaccia). Essa può essere utilizzata per visualizzare la configurazione corrente su un'interfaccia di rete. Un utente privilegiato può utilizzarla anche per modificare qualsiasi parametro su un'interfaccia di rete, sia essa una scheda Ethernet, un link seriale PPP o l'interfaccia di loopback. Per esempio, per mostrare la configurazione di tutte le interfacce di rete sull'host, possiamo utilizzare l'espressione:

```
$ /sbin/ifconfig -a
```

Su Windows, possiamo utilizzare l'analogo comando `ipconfig`:

```
> ipconfig /all
```

Su un host con un'interfaccia Ethernet si otterrà un output simile al seguente, da cui risulta la presenza di due interfacce:

```
eth0      Link encap:Ethernet  HWaddr 00:10:A4:FE:09:68
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.128
          inet6 addr: fe80::910:a4ff:ae0e:9a8/10 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MTU:1500 Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:9109 (8.8 Kb)  TX bytes:5658 (5.5 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:1540 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1540 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:231276 (225.8 Kb)  TX bytes:231276 (225.8 Kb)

lo:1      Link encap:Local Loopback
          inet addr:192.168.1.131  Mask:255.255.255.128
          UP LOOPBACK RUNNING MTU:16436 Metric:1

lo:2      Link encap:Local Loopback
          inet addr:192.168.1.132  Mask:255.255.255.128
          UP LOOPBACK RUNNING MTU:16436 Metric:1
```

La prima interfaccia è una scheda Ethernet, con il suo indirizzo fisso Ethernet assegnato dal produttore, più un indirizzo IP e una maschera di rete, entrambi configurabili. Questa particolare interfaccia si trova su un server supportato dall'IPv6, pertanto possiede entrambi gli indirizzi IPv4 e IPv6 assegnati dal sistema operativo. L'indirizzo IPv4 ha anch'esso una maschera di rete che lo colloca in una rete di classe C e un indirizzo broadcast che risulta dalla combinazione di indirizzo IP e maschera di rete. `ifconfig` comunica inoltre che l'interfaccia funziona ed è in grado di eseguire il broadcast, nonché di fornire un insieme di dati statistici sull'attività dell'interfaccia.

Il valore di MTU (Maximum Transmission Unit) è di 1500, il massimo consentito da Ethernet.

La seconda è l'interfaccia locale di loopback. Visto che si tratta di un dispositivo di loopback che non dipende dalla presenza di un reale hardware, esso non possiede né indirizzo Ethernet né indirizzo broadcast. Poiché il limite dei pacchetti di Ethernet non riguarda l'interfaccia di lo-

opback, essa può inviare pacchetti per un massimo di 16.436 byte. Poiché tutti i dati devono eseguire il loopback, la quantità ricevuta corrisponde esattamente a quella inviata.

La terza e la quarta interfaccia sono alias IP, ovvero funzionalità di alcuni moderni sistemi operativi che permettono l'assegnazione di diversi indirizzi IP alla stessa interfaccia, per produrre interfacce virtuali. Questi particolari alias sono destinati all'indirizzo di loopback, ma potremmo ugualmente ottenere un alias per l'interfaccia di Ethernet, se vogliamo rispondere a diversi indirizzi IP esterni sullo stesso server.

Si noti che gli indirizzi non devono necessariamente essere collegati all'indirizzo dell'interfaccia primaria, infatti queste interfacce hanno gli indirizzi sulla stessa classe di rete C dell'interfaccia Ethernet. Dato che per definizione sono su reti diverse, la maschera di rete è impostata in modo tale che un valore dell'ottetto finale di 0-127 è considerato separato da 128-255. Poiché le interfacce con alias sono rispettivamente la 131 e la 132, esse sono separate dall'interfaccia Ethernet, che ha un ottetto finale di 1.

Ovviamente, gli argomenti della riga di comando e l'output di `ifconfig` possono variare da sistema a sistema. Utilizzare `man ifconfig` per richiamare la pagina del manuale di `ifconfig` sul proprio sistema.

netstat

Assieme a `ifconfig`, `netstat` è un altro strumento standard di UNIX e uno dei più utili per il monitoraggio di una rete funzionante con UNIX. Esso è in grado di estrarre molti tipi diversi di informazioni su tutte o su una sola delle interfacce di rete. Una breve descrizione di alcuni degli argomenti di `netstat` (anche in questo caso, gli argomenti variano da un sistema all'altro) ci darà un'idea:

Argomento	Effetto
(nessuno)	Visualizza le connessioni aperte (socket)
-a	Mostra anche i socket di ascolto e di non ascolto
-c	Rivisualizza continuamente le tabelle selezionate
-i	Visualizza le interfacce di rete
-n	Visualizza gli indirizzi IP, non risolve i nomi
-r	Visualizza gli itinerari di rete
-s	Visualizza i dati statistici di rete
-v	Fornisce informazioni dettagliate

`netstat` supporta molti altri argomenti oltre a questi, soprattutto per la tabella di default (connessioni di rete aperte). Per maggiori dettagli, consultate la pagina del manuale (http://snowwhite.cis.uoguelph.ca/course_info/27420/netstat.html).

snoop e tcpdump

Tali utility permettono all'amministratore di esaminare i pacchetti inviati su una rete. `snoop` è disponibile su Solaris, mentre `tcpdump` è uno strumento gratuito dalle analoghe capacità disponibile su Linux e FreeBSD (sebbene possa essere utilizzato su qualsiasi piattaforma su cui possa essere costruito, visto che il codice sorgente è accessibile liberamente).

Ogni strumento permette di esaminare i pacchetti così come appaiono in rete. Varie opzioni permettono di filtrare i pacchetti secondo l'indirizzo IP e la porta di origine, l'indirizzo IP e la porta di destinazione, il protocollo, il tipo di messaggio e così via. Per esempio, le comunicazioni di Apache possono essere controllate sulla porta 80, scomposte in pacchetti di dati.

Si noti che non è necessario essere su un server per fare tutto ciò. Qualsiasi computer collegato alla stessa rete del server può farlo, sebbene di solito UNIX, per ragioni di sicurezza, consenta solo a un utente privilegiato di "spiare" sulla rete.

ping

`ping`, lo strumento più semplice e maneggevole di tutti, invia un messaggio ICMP a un nome di host remoto o a un indirizzo IP per stabilire se è presente e raggiungibile e riporta il tempo impiegato per compiere il tragitto. La maggior parte delle versioni di `ping` permette quindi di sondare il server remoto a intervalli regolari: questa funzione è utile per prevenire problemi di time-out e disconnessione delle connessioni di rete.

spray

`spray` è una variante di `ping` denominata in vari modi diversi. Questa utility inonda un server di destinazione con pacchetti di `ping` per verificare la capacità della rete e del server. Maggiore è la percentuale dei pacchetti che arrivano a destinazione, migliore è la rete. Tale sistema non è certamente gradito ai gestori di una rete di grande traffico, quindi dev'essere utilizzato con cautela.

traceroute

`traceroute` è utile per diagnosticare i problemi nello stabilire le connessioni di rete, per esempio nei casi in cui `ping` non riesce a raggiungere il server remoto. Essa utilizza il protocollo ICMP per richiedere informazioni a ogni passo intermedio dell'itinerario, dall'host al destinatario. In Internet, può fornire in alcuni casi un output di oltre venti righe.

`traceroute` risulta particolarmente utile nel diagnosticare i problemi legati alle connessioni fallite, poiché talvolta riesce a localizzare il punto in cui il tentativo di connessione fallisce. Può quindi essere utile per determinare i sistemi configurati in modo sbagliato o difettoso nella rete. Per maggiori informazioni, consigliamo anche in questo caso di consultare la pagina del manuale (<http://www.stopspam.org/usenet/mmf/man/traceroute.html>).

Scelta dell'hardware del server

Se vogliamo acquistare l'hardware per il nostro sito Web, dobbiamo considerare varie questioni, soprattutto quella principale, se acquistarlo o meno. Consultate il paragrafo "Che lo faccia qualcun altro" alla fine di questo capitolo.

Piattaforme supportate

Apache funziona su numerose piattaforme. Solitamente funziona su sistemi UNIX, tra cui i più noti sono Solaris e i sistemi operativi gratuiti simili a UNIX, Linux e FreeBSD. Anche MacOS X è popolare, se non altro perché ogni computer contiene un'installazione Apache di default.

Apache può essere eseguito anche su Windows NT, sebbene il funzionamento di Apache 1.3 non sia tanto scorrevole quanto l'implementazione di UNIX. Apache 2.0 è più adatto a Windows e fornisce migliori prestazioni e affidabilità. Si sta inoltre cercando di importare Apache su altre piattaforme dove non è ancora supportato. Tuttavia, c'è un unico motivo ragionevole per scegliere una di queste piattaforme: possedere già l'hardware relativo.

Le società che hanno stipulato contratti di assistenza e considerano questo aspetto prioritario, dovranno probabilmente optare per le piattaforme più importanti, posto che funzionino con Apache e non presentino problemi di prestazioni e stabilità. Chi ha problemi di budget potrà optare per un PC economico con Linux o FreeBSD; si tratta di una soluzione poco costosa che presenta il vantaggio di una buona stabilità su entrambe le piattaforme. La costruzione di un cluster economico da una selezione di server economici è una soluzione più pratica di quanto potrebbe sembrare inizialmente. Possiamo eseguire un semplice clustering utilizzando nient'altro che Apache e un server dei nomi.

Per un server semplice e siti Web senza particolari esigenze basterebbe persino un vecchio 486. Se abbiamo del vecchio hardware da parte e vogliamo posticipare la decisione di acquistare un nuovo PC finché non avremo le idee più chiare su quello che ci serve, i vecchi computer inu-

tilizzati possono andare benissimo. In alternativa, possiamo comprare un PC economico per sviluppare una soluzione provvisoria. Rimandare l'acquisto il più possibile consentirà di acquistare in seguito una macchina più potente a un prezzo inferiore.

Nell'ambito del software gratuito, Linux e FreeBSD sono entrambe scelte diffuse. La differenza principale tra di esse è che FreeBSD è leggermente più stabile e ha un supporto di networking più rapido, mentre per Linux c'è molto più software disponibile. La differenza è minima, comunque, poiché Linux è sufficientemente stabile per la maggior parte delle applicazioni Web, al contrario l'importazione di software da Linux a FreeBSD è di solito un'operazione relativamente semplice.

Se la stabilità è di importanza vitale e non intendiamo installare molto software aggiuntivo, sceglieremo FreeBSD. Se progettiamo di installare pacchetti aggiuntivi per il supporto dei database, la sicurezza o l'e-commerce, Linux è probabilmente da preferire. Altre varianti BSD note per i server di Web sono OpenBSD, NetBSD e ovviamente MacOS X.

Nel momento in cui scriviamo, le seguenti piattaforme supportano completamente Apache:

AIX	A/UX	BS2000/OSD
BSDI	DGUX	DigitalUNIX
FreeBSD	HP-UX	IRIX
Linux	MacOS X	NetBSD
Netware	OpenBSD	OS/2
OSF/1	QNX	ReliantUNIX
SINIX	Solaris	SunOS
UNIXWare	Windows 9x e ME	Windows NT, 2000 e XP

Requisiti fondamentali del server

Se ci troviamo in un ambiente omogeneo com'è quello di un'azienda, è ragionevole utilizzare lo stesso tipo di attrezzatura, se non altro per salvaguardare la salute mentale degli amministratori di sistema e per rendere più semplice l'amministrazione della rete.

Tuttavia, questo non è un punto così importante come potrebbe sembrare. Se il nostro server non è adeguatamente collegato al resto dell'intranet aziendale (per esempio, non deve accedere a un database), è una buona idea isolare il server dalla nostra intranet a scopi di sicurezza. Poiché non esiste comunicazione tra il server del Web e gli altri server, non nasceranno neanche problemi di compatibilità.

Apache funzionerà quasi su ogni macchina; pertanto, a meno che non abbiamo una ragione specifica per scegliere una determinata marca di hardware, qualsiasi PC di costo medio-basso andrà bene, purché sia affidabile. La stabilità è molto più importante del marchio.

Funzionamento del server su hardware dedicato

Benché si tratti probabilmente di un'ovvietà, vale la pena di accennare alla possibilità di eseguire Apache sul suo hardware dedicato. Dato il carico notevole imposto da un server di rete alla CPU, al disco e alla rete, e dato che Apache funziona anche su un hardware molto economico, non ci sono ragioni per non acquistare un server dedicato per i siti Web evitando di condividere risorse con altre applicazioni. Inoltre, non è una buona idea utilizzare un computer che ospita applicazioni importanti e al contempo file di un sito Web di pubblico accesso.

Server ad alte prestazioni/alta affidabilità

Per le applicazioni complesse si può prendere in considerazione l'idea di utilizzare un sistema a più processori. Con i sistemi espansibili possiamo aumentare progressivamente la potenza del server installandovi altri processori o memoria aggiuntiva, man mano che se ne presenta l'esigenza.

In alternativa, è anche possibile creare un cluster di varie macchine indipendenti per ottenere un singolo server virtuale (questa soluzione è da preferire dal punto di vista delle spese e dell'affidabilità). Esistono vari modi per farlo, oltre alla soluzione dei cluster personalizzati di cui parleremo nel Capitolo 7.

Memoria

La memoria non è mai abbastanza: più memoria abbiamo a disposizione, più dati potranno essere registrati nella cache per un accesso rapido. Questo vale non solo per Apache ma anche per tutti gli altri processi eseguiti sul server, come le applicazioni database.

Più precisamente, la quantità di memoria che ci serve è pari a quella che permette al server e a tutti i processi attivi di funzionare senza utilizzare la memoria virtuale. Se il sistema operativo finisce per essere a corto di memoria, dovrà temporaneamente spostare i dati dalla memoria al disco (cosiddetto **swapping**). Quando i dati sono nuovamente necessari, essi dovranno essere letti dal disco con contemporaneo swapping di altri dati contenuti nella memoria virtuale, a meno che, nel frattempo, non si sia liberata una quantità opportuna di tale memoria.

È chiaro che non si tratta di un sistema efficiente: esso rallenta il processo in cui sono necessari tali dati e blocca il disco e il processore, costringendoli a smistare i dati invece di servire le pagine Web. Se i dati che vengono travasati sono rappresentati dalla cache del server Web o da tabelle di database di frequente accesso, l'impatto sulle prestazioni può essere rilevante.

Per calcolare la quantità di memoria che ci serve, occorre sommare le quantità di memoria necessarie a ogni applicazione. Poiché si tratta di una scienza inesatta, vale la seguente regola empirica: "aggiungere più memoria". In ultima analisi, solo l'osservazione del comportamento del server mentre è in funzione ci dirà se abbiamo sufficiente memoria o meno.

Lo strumento `vmstat` sulla maggior parte dei sistemi UNIX è un modo per monitorare in che misura il server sovraccarica la memoria e il tempo che dedica allo swapping. Strumenti simili sono disponibili anche per altre piattaforme. In Windows NT è presente un ottimo strumento, chiamato `perfmon` (*performance monitor*).

È importante, inoltre, possedere un sistema operativo che gestisca la memoria in modo efficiente (consultate il paragrafo "Lista di controllo del sistema operativo" in questo capitolo).

Interfaccia di rete

Le prestazioni della CPU e un'abbondanza di memoria da sole non bastano e porteranno frequentemente alla formazione di colli di bottiglia a causa di una velocità di I/O insufficiente del sistema (frequenza di accesso alla scheda dell'interfaccia e al disco fisso).

In un'intranet, la rete e la scheda dell'interfaccia sono molto esigenti e una normale connessione 10Base2 o 10BaseT può facilmente diventare un problema. Una rete 10Base può funzionare con una velocità massima di trasmissione dati di 6-8 megabit al secondo, ma un server di rete a cui si accede con un tasso di 90 hit al secondo raggiungerà presto tale valore.

Le schede di rete 100baseT e il relativo cablaggio sono divenuti decisamente più economici rispetto al passato, per cui non c'è alcun motivo di investire in reti a 10Base a meno che non possediamo già una rete 10Base di tipo tradizionale. Anche in tal caso, le schede di rete duali 10/100Base sono l'opzione migliore, perché possiamo aggiornare successivamente il resto della rete. Per le applicazioni più complesse è disponibile anche Gigabyte Ethernet, sebbene sia molto più costoso da implementare.

Purché gli altri computer non sollecitino indebitamente la rete, l'uso di una normale scheda Ethernet sarà sufficiente nella maggior parte dei casi, a condizione che non si tratti della scheda più economica disponibile in offerta speciale in un grande catena di prodotti informatici. Si noti che molte schede a basso prezzo non utilizzano funzionalità quali il DMA (*Direct Memory Access*); le schede con questa capacità vengono identificate come in grado di effettuare il *bus-mastering*, quindi hanno prestazioni sensibilmente inferiori, sebbene siano compatibili con quelle più costose.

Connessioni di rete duali

Una soluzione eccellente, soprattutto se intendiamo collegare i server a un ISP per la connessione a Internet e anche a un'intranet locale, è quella di inserire due schede di rete e assegnare loro differenti indirizzi IP su reti differenti. L'interfaccia di rete esterna è quindi utilizzata esclusivamente per l'accesso al server di rete, mentre l'interfaccia di rete interna si collega al server dei database o ai sistemi di backup, permettendoci di elaborare le richieste del database e di eseguire i backup senza alcun impatto sulla larghezza di banda esterna. Analogamente, un sito Web con un intenso traffico non pregiudicherà la larghezza di banda sulla rete interna.

Le interfacce di rete duali hanno anche il vantaggio di una maggiore sicurezza: isolando la rete interna da quella esterna ed eliminando qualsiasi routing tra di esse, diviene relativamente semplice negare agli utenti esterni l'accesso alla rete interna. Per esempio, se abbiamo un firewall, possiamo inserirlo tra l'interfaccia di rete interna e il resto della rete, escludendo il server fuori ma includendo tutto il resto.

Connessione a Internet

Se il server dev'essere connesso a Internet, è necessario considerare attentamente il tipo di connessione che utilizzeremo e le capacità del provider che ce la fornisce.

Prima di scegliere un provider bisogna porsi alcune domande.

- ☐ È affidabile?
- ☐ Possiede una buona connettività a Internet (a quale livello è, ha circuiti ridondanti)?
- ☐ Dobbiamo condividere la larghezza di banda con molti altri clienti?
- ☐ In tal caso, il provider offre una connessione dedicata?

Se abbiamo un sito a carattere internazionale (per esempio, uno contenente tutti i siti regionali di una società internazionale in un'unica posizione), dobbiamo porci un'ulteriore domanda:

- ☐ Possiede una buona connettività globale?

Infine, per valutare bene il tipo di assistenza che il provider offre, dobbiamo chiederci:

- ☐ L'assistenza tecnica fornisce risposte pertinenti quando ci rivolgiamo a essa?

Se un provider offre una grande larghezza di banda, ciò non significa necessariamente che gli utenti di Internet possano farne uso: tutto dipende dalla qualità della connessione del provider ai suoi pari e della dorsale Internet in generale. Molti provider si basano su un solo fornitore per la loro connettività e, se tale fornitore è sovraccarico, la grande larghezza di banda sarà inutile per noi e i nostri visitatori, anche se la larghezza di banda in uscita del provider in teoria è più che sufficiente.

Disco fisso e controller

I dischi fissi veloci e un adeguato controller sono una scelta sensata per un server di rete e un sistema SCSI è di gran lunga preferibile a un sistema IDE se le prestazioni rappresentano un aspetto importante.

Per i siti Web frequentemente visitati, è importante anche utilizzare vari dischi più piccoli invece che un unico grande disco fisso. Se, per esempio, occorre gestire un database o vari server virtuali di grandi dimensioni, è consigliabile memorizzare i dati su diversi dischi per garantire prestazioni di accesso superiori, dal momento che un unico disco fisso può essere letto solo una posizione per volta.

Si può anche utilizzare il RAID 0 per aumentare le prestazioni di un array di dischi. Combinato con RAID 1, che assicura la ridondanza, può costituire un modo molto efficace per migliorare le prestazioni del server. Questo hardware è anche noto come RAID 0+1, RAID 1+0 e RAID 10, ma si tratta sempre dello stesso. Tuttavia, questa soluzione può essere costosa.

Lista di controllo del sistema operativo

Affinché il server funzioni efficacemente (cioè affinché sia stabile e al contempo efficiente), il sistema operativo che lo ospita deve riuscire a svolgere tutte le operazioni richieste. Abbiamo discusso dei sistemi operativi in relazione alle piattaforme supportate da Apache e abbiamo detto che UNIX è la piattaforma di server preferita per le installazioni di Apache. Qualsiasi sistema operativo scegliamo, dovrebbe possedere tutte le caratteristiche seguenti.

- ❑ **Stabilità**
Il sistema operativo dev'essere affidabile e in grado di funzionare indefinitamente senza che si debba riavviarlo. Una cattiva gestione della memoria è una delle cause maggiori di inaffidabilità a lungo termine.
- ❑ **Sicurezza**
Il sistema operativo dovrebbe resistere a qualsiasi tipo di attacco, tra cui gli attacchi denial-of-service (in cui si cerca di esaurire le risorse del sistema, impedendo agli utenti legittimi di accedere al servizio), e avere un buon indice di sicurezza. Le falle che vengono scoperte nella sicurezza dovrebbero essere riparate rapidamente dal responsabile. Si noti che con "rapidamente" si intende nel giro di qualche giorno, non di settimane.
- ❑ **Prestazioni**
Il sistema operativo dovrebbe utilizzare le risorse in modo efficace, gestendo la rete senza pesare indebitamente sul resto del sistema operativo ed eseguendo in modo efficace il passaggio da un task all'altro. Apache esegue processi multipli per gestire le connessioni in arrivo; se il passaggio da un processo all'altro è inefficiente, questo può causare una perdita di prestazioni. Se progettiamo di utilizzare un sistema a più processori, è importante tenere presente anche le prestazioni del processore simmetrico (SMP).
- ❑ **Manutenzione**
Il sistema operativo dovrebbe essere semplice da aggiornare o da correggere con patch per motivi di sicurezza; non dovrebbe essere necessario eseguire il reboot o scollegarlo se non per importanti aggiornamenti e non dovrebbe essere necessario eseguire il reboot dell'intero sistema per la manutenzione o l'aggiornamento di una singola parte di esso.
- ❑ **Memoria**
Il sistema operativo dovrebbe utilizzare in modo efficiente la memoria, evitando lo swapping a meno che non sia assolutamente necessario, facendolo comunque in modo intelligente, e non dovrebbe essere soggetto a perdite di dati dalla memoria con conseguenti blocchi inutili della memoria stessa. (Il software leaky, ossia che tende a perdere dati, è una delle principali cause di inaffidabilità dei server di rete. Per esempio, fino a poco tempo fa, Windows NT era disastroso da questo punto di vista). Tuttavia, le applicazioni leaky sono anche problematiche. Fortunatamente, Apache non è una di esse, sebbene sia stata in passato meno brillante di quanto sia ora.

I moduli di terze parti possono provocare problemi, ma Apache fornisce la direttiva `MaxRequestsPerChild` che periodicamente attua un riavvio forzato dei processi di Apache, impedendo ai moduli "indisciplinati" di comportarsi troppo male. Se prevediamo di utilizzare grandi applicazioni come i server di gestione dei database, dobbiamo anche controllare le loro prestazioni.

Ridondanza e backup

Se progettiamo di utilizzare un server di una certa importanza, dobbiamo prestare una certa attenzione a come intendiamo ripristinarlo, se per qualche ragione smettesse di funzionare. Potremmo per esempio avere un guasto dell'hardware, oppure potremmo subire un attacco di hacker e un danno ai dati. Un array RAID è una buona prima linea di difesa, ma può anche risultare costoso. Esso assicura il backup nel server, il che non è molto comodo, se il server dovesse essere coinvolto in un incendio o un'esplosione (può succedere).

Una semplice soluzione di backup consiste nell'equipaggiare il server con un lettore DAT o un altro dispositivo di memorizzazione e nel configurare il server in modo che copi automatica-

mente i file importanti su un nastro magnetico a intervalli regolari e prestabiliti. Un tale sistema può essere attivato anche senza un software di backup speciale; su una piattaforma UNIX, la semplice utility `cron` consentirà di farlo automaticamente.

Una soluzione migliore consiste nel compiere il backup attraverso una rete interna, se esiste. Questo consentirebbe di copiare i dati su un server di backup esterno, che potrebbe sostituire il server principale in caso di guasti. Questa soluzione renderebbe inoltre superfluo l'intervento manuale, visto che i nastri DAT non vengono scambiati tra loro automaticamente.

Se il nostro server è su Internet (ma anche nel caso contrario), dobbiamo prendere le opportune precauzioni perché anch'esso non venga danneggiato. Qualora questo accada, c'è solo un'operazione da fare: reinstallare tutto a partire da copie di backup affidabili; questo comprende la reinstallazione del sistema operativo, la sua riconfigurazione e la reinstallazione del sito o dei siti dalle copie di backup. Se facciamo una copia di backup sullo stesso supporto ogni giorno e ci accorgiamo che c'è un problema solo dopo aver effettuato l'indomani il successivo backup, non avremo alcun backup affidabile. La morale è che bisogna effettuare backup multipli e applicare un'etichetta con la data.

Esistono in commercio vari strumenti per il backup della rete e la nostra scelta può essere influenzata dall'ambiente del server: la strategia di backup aziendale probabilmente si può estendere anche al server. Le opzioni gratuite comprendono strumenti poco elaborati, come l'FTP o l'NFS, per copiare gli alberi delle directory da un server all'altro (a meno che non abbiamo un'ottima ragione per farlo, non dovremmo attivare neanche l'NFS sul server, poiché questo può comprometterne la sicurezza).

Un tool gratuito migliore per i backup è `rsync`, una versione intelligente del comando standard di UNIX `rcp` (*remote copy*), che copia solo le differenze tra le gerarchie di directory. Meglio ancora, esso può funzionare attraverso una connessione crittografata con `ssh` (*secure shell*), un altro strumento gratuito. Se abbiamo l'esigenza di effettuare backup remoti dei file del server tramite Internet, è il caso di considerare seriamente questo approccio. (Parleremo di `rsync` e `ssh` nel Capitolo 10.) Sempre a proposito dei tool gratuiti, occorre accennare a un'altra opzione più avanzata, cioè il CVS (*Concurrent Versioning System*). Applicato soprattutto al codice sorgente, funziona bene anche con i file HTML. Per maggiori dettagli, consultate <http://www.cvshome.org/>.

Una considerazione finale sui backup effettuati tramite la rete: anche se utilizziamo un sistema di backup intelligente in grado di eseguire il backup incrementale del server, un sito molto grande può significare, comunque, il trasferimento di una grande quantità di dati. Se poi il server è molto occupato, questi dati, ogni volta che viene eseguito un backup, consumeranno una larghezza di banda che altrimenti potrebbe essere utilizzata per gestire le richieste del browser; pertanto è necessario pianificare i backup nei momenti giusti. Se abbiamo molti dati da copiare, prendiamo in considerazione l'ipotesi di farlo a stadi (per esempio suddividendo il backup per directory), e farlo sicuramente in modo incrementale. Con connessioni di rete duali e un backup eseguito sulla connessione interna, lasciando quella esterna a disposizione delle richieste HTTP, si ottengono vantaggi sicuri.

Soluzioni specifiche di hardware

Diversi produttori attualmente mettono in commercio hardware con Apache o un suo derivato preinstallato, insieme a software di amministrazione per semplificare la configurazione e la manutenzione del server. A questo punto, tutte le soluzioni sono basate su UNIX, prevalentemente Linux. Anche vari provider offrono alcune di queste soluzioni come server dedicati da acquistare o noleggiare.

Sun Cobalt Qube e RaQ

Cobalt fu una delle prime società a progettare hardware specificamente destinato a Linux, e la gamma di microserver dal tipico colore blu cobalto è divenuta molto popolare fra le aziende e i provider.

Cobalt, attualmente di proprietà di Sun Microsystems, offre due prodotti di base: Qube 3, destinato ad aziende con limitate esigenze di hardware, è un vero e proprio cubo, mentre RaQ è

un server montato su rack e destinato ai provider che vogliono installare varie decine di server in un solo rack. I due prodotti sono molto simili, ma non identici dal punto di vista del software.

Le versioni meno recenti erano basate su un processore MIPS; quelle più recenti utilizzano processori Intel. Entrambi usano una versione abbreviata di Linux Red Hat, ed entrambi i server, Cobalt Qube e Cobalt RaQ, sono associati a uno strumento di configurazione basato su Web che permette al server di essere configurato a distanza: un grande vantaggio per gli amministratori di sistema. Le configurazioni di siti Web, host virtuali e DNS possono essere tutte eseguite tramite un browser e su Cobalt girano intelligentemente due copie di Apache in simultanea, una per servire i siti Web e una per configurarli. Così, anche se roviniamo la configurazione principale di Apache, il server Apache di amministrazione dovrebbe continuare a funzionare.

Un problema connesso ai server più vecchi, che si trovano spesso di seconda mano a prezzi convenienti, è dovuto al fatto che non utilizzano una comune architettura del processore e per i processori MIPS sono disponibili relativamente pochi software in forma binaria, rispetto ai derivati Intel. Di conseguenza, questi server sono adatti per applicazioni che non richiedono estese installazioni di software aggiuntivo. Se intendiamo installare molte applicazioni di terze parti, dovremo spesso essere costretti a costruirle noi stessi dal sorgente. Poiché Red Hat utilizza il formato di pacchetti RPM, abbiamo un alleato nell'archivio RPM <http://www.rpmfind.net/>. Per le macchine più nuove, il problema ovviamente non sussiste.

Per maggiori dettagli, consultate <http://www.cobalt.com/>.

IBM

IBM offre una gamma di prodotti presentati con il server HTTPD di IBM, un Apache ritagliato i cui numeri di versione ricalcano fedelmente quelli di Apache. Esso si combina perfettamente con la gamma di prodotti WebSphere IBM, disponibili su piattaforme AIX, Solaris, Linux e Windows.

IBM offre inoltre pacchetti che comprendono JavaServer Pages e tecnologia Servlet con HTTPD. Quest'ultimo si basa su un altro progetto open-source della Apache Software Foundation: Jakarta Tomcat (<http://jakarta.apache.org/>). Si tratta di un'altra opzione di contenuto dinamico che sgombra il campo dalle questioni sull'integrazione discusse nel Capitolo 12, in cui esamineremo Tomcat in quanto estensione di Apache.

Per maggiori dettagli, consultate <http://www.ibm.com/>.

Linux generico + Apache

Linux è ora disponibile preinstallato in una serie di hardware per server destinati sia a usi di desktop sia ad applicazioni di server su Internet. Le installazioni standard di Linux non hanno la configurazione front-end di sistemi quali RaQ/Qube o Netwinder, ma prevedono l'assistenza per chi ne ha bisogno. La lista dei produttori aumenta di continuo: la pagina Web Linux VAR HOWTO (<http://www.linuxdoc.org/HOWTO/VAR-HOWTO.html>) e altre pagine contengono alcune utili indicazioni.

Che lo faccia qualcun altro

Esiste un'alternativa al dover impostare da soli il server, affrontando tutte le relative questioni di affidabilità, connettività e backup: è possibile acquistare o noleggiare un server dedicato da un provider. Tale possibilità viene comunemente denominata "colocazione".

I vantaggi sono costituiti dal fatto che il provider gestisce tutte le questioni collegate alla manutenzione ordinaria, lasciando a noi tutta la flessibilità di un server che ci appartiene completamente. Possiamo persino ricostruire e riconfigurare Apache come preferiamo, dal momento che abbiamo il controllo totale del server. Questo significa che abbiamo anche la possibilità di distruggere il server, pertanto il fatto che il server non sia fisicamente presente non elimina la necessità di un amministratore di rete.

Lo svantaggio consiste nella lontananza fisica dal server. Se insorge un grave problema, potremmo non essere in grado di accedervi per scoprire di che cosa si tratta. Molto probabilm-

te, l'ISP imporrà restrizioni alla larghezza di banda, e questo va tenuto presente. Inoltre dipendiamo dall'assistenza del provider, per cui è meglio verificare la qualità del servizio offerto prima di firmare il contratto.

Si noti che i servizi variano da un provider all'altro: alcuni eseguono automaticamente il backup dei file del server, mentre altri non lo prevedono. Come per la maggior parte dei problemi relativi a Internet, vale la pena controllare i potenziali provider consultando le liste di discussione e i newsgroup Usenet. *Caveat Emptor!*

Per un'ulteriore documentazione introduttiva, consultate:

<http://httpd.apache.org/docs/misc/FAQ.html#what>.

Riepilogo

In questo capitolo abbiamo esaminato i concetti fondamentali relativi alla natura dei server di rete e in particolare abbiamo acquisito le nozioni di base su Apache. Ci sono molte ragioni per cui Apache è un server Web tanto diffuso, tra cui il fatto importante che è gratuito. Il modo migliore di ottenere assistenza per Apache consiste nell'utilizzare il supporto informativo e informale della comunità on-line, molto attiva riguardo al suo sviluppo e alla sua manutenzione.

Abbiamo visto come funziona Apache su UNIX e Windows. L'amministrazione di una rete è un compito complesso e abbiamo dato uno sguardo ad alcuni tool quali `ifconfig`, `netstat`, `snoop` e `tcpdump`, `ping`, `spray` e `traceroute`. Nell'ultima parte del capitolo abbiamo preso in esame i requisiti fondamentali del server e alcune soluzioni specifiche di hardware per il nostro server Web.

Nel prossimo capitolo ci occuperemo dell'installazione e della configurazione di Apache come server Web di base.