UNIVERSITA' DEGLI SUDI DI PAVIA
FACOLTA' DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica
Sede di Mantova

Sviluppo di un sito web per il Laboratorio Reti su stack multiprotocollo IPv4-IPv6

Relatore:
Prof. GIUSEPPE FEDERICO ROSSI

Tesi di laurea di:
DARIO GENOVA

Anno Accademico 2004/2005

Indice dei contenuti

Introduzione

La rete internet, a tutti familiare, si avvale, da oltre 20 anni, di un protocollo di comunicazione noto come Internet protocol version 4. Brevemente IPv4.

L'idea di una rete di comunicazioni in grado di connettere tra loro computer anche molto distanti iniziò a farsi strada negli anni 50. A quell'epoca risalgono i primi studi relativi alle reti decentralizzate, alla teoria delle code ed alla commutazione di pacchetto.

La prima realizzazione pratica di tale idea, il progetto ARPANET, vide la luce nel 1969. Nel dicembre del 1969 ARPANET collegava tra loro quattro IMP: interface message processors. Gli IMP erano piccoli computer cui era deputata la commutazione dei pacchetti. Essi erano interconnessi tra loro attraverso modem a 50 kbit/secondo; erano connessi a host locali tramite interfacce seriali. Ogni IMP avrebbe potuto collegarsi con altri 5, tramite linee di comunicazione dedicate, e servire 4 host. Totale: 24 nodi.

La prima rete TCP/IP divenne operativa nel 1984. In questi 20 anni la rete è passata dal servire un pubblico eminentemente tecnico/scientifico ad uno generico ed assai più vasto, che la utilizza per scopi commerciali, politici, ideologici, d'intrattenimento. IPv4 connette oggi fra loro circa 400 milioni di nodi, ed il numero cresce sempre più rapidamente. Quanto a lungo può continuare a farlo?

Già negli anni 90 ci si pose la domanda. Si rispose con IPv6.

Questo elaborato mira a sperimentare una delle soluzioni proposte al problema della transizione da IPv4 ad IPv6.

Sezione I

Introduzione alle tecnologie

Capitolo I

La transizione verso Ipv6

Il meccanismo Dual Stack

Ipv6

Ipv6, acronimo di Internet protocol version 6, è uno standard utilizzato dalle apparecchiature elettroniche per lo scambio di informazioni attraverso una rete di comunicazione a commutazione di pacchetto, situato a livello "network" nella pila ISO-OSI. E' la seconda versione di protocollo internet ad essere adottata ufficialmente dopo Ipv4 (il progetto Ipv5, che prevedeva spazi di indirizzamento a 64 bit e supporto per funzioni di streaming, venne reputato una evoluzione troppo costosa, a fronte dei ridotti benefici, per essere seriamente preso in considerazione per il rilascio pubblico).

Il progetto Ipv6 venne concepito inizialmente come risposta al problema dell'esaurimento dello spazio di indirizzamento Ipv4: 32 bit limitano a poco più di quattro miliardi gli indirizzi unici utilizzabili, meno di uno per ogni abitante del pianeta. L'introduzione della tecnologia NAT ha temporaneamente alleviato il problema, ma nella prospettiva di dotare ogni dispositivo elettronico mobile (telefoni cellulari, pda…) di un indirizzo IP per l'accesso ad internet il protocollo Ipv4 mostra tutti i propri limiti. L'introduzione dello spazio di indirizzamento a 128 bit di Ipv6 permetterà l'assegnazione di circa 3.4×10^{38} indirizzi.

Ipv6 però va ben oltre questo: è stato concepito per fornire mobilità ai dispositivi (nomadic computing). Prevede la possibilità di configurare automaticamente connessioni di rete ed il routing di un gran numero di dispositivi, controllo della "qualità di servizio" per le applicazioni che hanno particolari necessità, meccanismi per la tutela della privacy, scopi per i quali Ipv4 si dimostra inadeguato.

Indirizzamento a 128 bit

Come già affermato, il numero di indirizzi che Ipv6 può fornire è enorme (15 ordini di grandezza oltre il numero di Avogadro!). La ragione di questa scelta, apparentemente eccessiva, è da ricercarsi non tanto nel timore di una nuova penuria di indirizzi, quanto nella desiderabilità di avere uno spazio degli indirizzi il meno frammentato possibile, introducendo una gerarchia di indirizzamento. Questa scelta riduce il numero degli indirizzi effettivamente disponibili, ma incrementa notevolmente l'efficienza del protocollo, riducendo l'overhead dovuto al routing.

Tipicamente gli indirizzi ipv6 sono suddivisi in due parti: 64 bit dedicati al prefisso di rete ed i rimanenti per l'identificazione degli host.

E' possibile classificare gli indirizzi ipv6 in tre gruppi principali: unicast, anycast e multicast. La configurazione dei bit iniziali dell'indirizzo ne specifica il tipo.
Ogni interfaccia collegata ad una rete Ipv6 possiede uno o più indirizzi *unicast*, ed ogni nodo di una rete può possedere una o più interfacce. Si può dunque dire che ogni nodo è identificato da uno qualsiasi degli indirizzi unicast appartenenti alle proprie interfacce. Gli indirizzi di tipo *anycast* identificano invece un gruppo di interfacce. Un pacchetto inoltrato ad un indirizzo di questo tipo verrà recapitato ad una qualsiasi delle interfacce appartenenti al gruppo, tipicamente la più "vicina" secondo i criteri di instradamento dei router intermedi. Gli indirizzi *multicast* similmente identificano gruppi di interfacce. Tuttavia un pacchetto inviato ad un indirizzo multicast raggiungerà tutte le interfacce del gruppo. Non è invece previsto un indirizzo di tipo broadcast.

Alcuni particolari tipi di indirizzo unicast degni di nota sono:

- *Provider based unicast addresses*, utilizzati per le comunicazioni su scala globale. Funzionano similmente agli indirizzi Ipv4 con CIDR.
- *Link local use addresses*, che sono visibili solo sul link fisico locale e possono o meno essere unici a livello globale. Sono solitamente generati in modo automatico alla connessione dell'interfaccia.

E' possibile convertire indirizzi Ipv4 in indirizzi Ipv6: in questo caso i primi 80 bit vengono impostati a 0. I successivi 16 vengono impostati a 0 se l'host identificato supporta Ipv6, ad 1 se l'host supporta solamente Ipv4. Gli ultimi 32 bit riflettono la configurazione binaria dell'indirizzo Ipv4 di partenza.

Pacchetti IPv6

Il pacchetto IPv6 è composto da 2 parti: header e payload.

L'*header* ha una lunghezza di 40 byte, di cui 32 riservati agli indirizzi di provenienza e destinazione ed i restanti suddivisi come illustrato in figura 1. Per quanto riguarda il significato dei campi restanti:

- Ver (4 bit) Indica la versione del protocollo.
- Prio (4 bit) Indica la priorità del pacchetto. Vedere oltre.
- Flow Label (24 bit) Contiene informazioni sul Quality of Service (QoS). Vedere oltre.
- Payload Length (16 bit) Lunghezza del Payload in ottetti.
- Next Hdr (8 bit) Identifica il tipo di header che segue l'header IPv6, importante vista l'introduzione degli header extensions per le opzioni.
- Hop Limit (8 bit) Determina il TTL del pacchetto

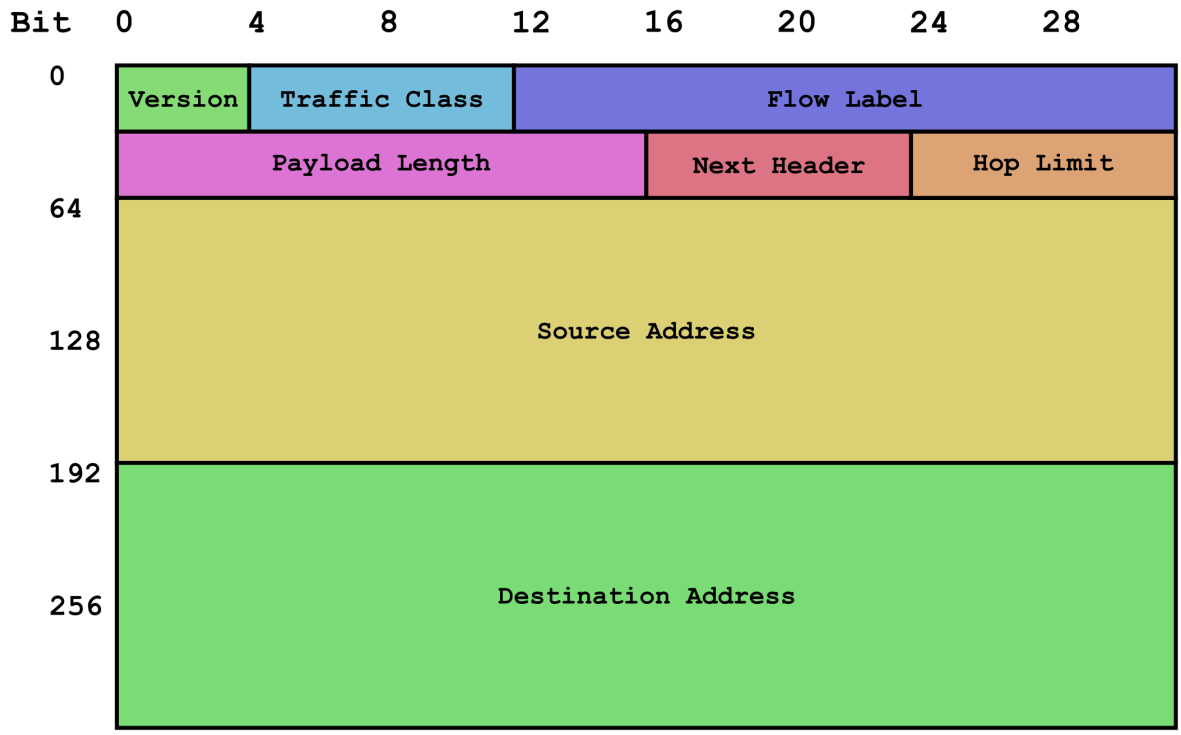

Figura 1 - Struttura dell'header di un pacchetto IPv6

IPv6 introduce un nuovo meccanismo per la gestione delle opzioni: esse vengono inserite in uno header specifico e separato, di lunghezza variabile, posto tra quello dello strato di trasporto e quello dello strato di rete, ciò che consente di includere nel datagram solo le informazioni realmente utili.

La maggior parte delle opzioni non viene esaminata dai router intermedi, contribuendo in questo modo ad alleggerire il carico di lavoro su di essi.
In secondo luogo, non esiste limite alle dimensioni dell'header per le opzioni: questo permette una serie di applicazioni impossibili da effettuare con IPv4.

Tra le opzioni correntemente definite ricordiamo:

- l'extended routing, che permette di specificare un certo numero di router per i quali si vuole che il datagram passi necessariamente
- frammentazione
- payload cifrato, che contiene informazioni relative alla sicurezza
- autenticazione, che consente di verificare l'identità del trasmittente

- opzioni Hop by Hop, che devono essere esaminate dai nodi intermedi
- opzioni di destinazione, per il solo nodo finale.

Anche il meccanismo di frammentazione ha subito modifiche: se la ricomposizione del datagram avviene ancora alla destinazione finale, non è più previsto che i router intermedi frammentino il traffico. La sorgente è responsabile della frammentazione, e può scegliere di utilizzare lo MTU predefinito di 1280 ottetti oppure sondare la MTU minima del percorso lungo cui il pacchetto deve essere instradato. Questo meccanismo, detto frammentazione punto-punto, consente da un lato di ridurre il carico di lavoro dei router intermedi (l'overhead dovuto alla frammentazione può essere molto pesante), ma contraddice una delle proprietà essenziali di IPv4: l'instradamento dei pacchetti non può più mutare dinamicamente. Infatti una mutazione del percorso può comportare una diversa MTU minima. E' stato introdotto per questo un nuovo messaggio di errore ICMP che i router inviano alla sorgente del traffico qualora si rende necessaria ulteriore frammentazione. La sorgente si farà allora carico di calcolare nuovamente la MTU per il nuovo percorso.

IPv6 e QoS

I campi flow-label e priorità permettono ad un host di identificare i pacchetti che richiedono un trattamento speciale in termini di banda o ritardo. E' una capacità importante per il supporto di applicazioni di tipo multimediale od in tempo reale.

Il campo *flow label*, è tuttora in fase sperimentale e potrebbe essere utilizzato dagli host sorgente per etichettare i datagram in modo da informare i router intermedi riguardo a particolari trattamenti da riservare loro. Host e router che non supportino questo tipo di servizio devono impostare il campo a 0 quando

generano un datagram, lasciarlo invariato quando lo inoltrano, ignorarlo quando ricevono. Si definisce *flow* (flusso) un insieme di pacchetti inviati da una medesima sorgente ad una medesima destinazione ed uniti dalla necessità di ricevere uno stesso speciale trattamento da parte dei router intermedi. La definizione del trattamento può essere già presente sui protocolli di controllo dei router, oppure può essere specificata nelle opzioni Hop by Hop. Possono esistere molteplici flussi tra una sorgente ed una destinazione, così come può coesistervi anche traffico non appartenente ad alcun flusso. Un flusso è identificato da una combinazione dell'indirizzo sorgente e da un numero generato in modo pseudo-casuale.

Il campo *priority* consente di identificare 16 diversi tipi di traffico di crescente priorità. I valori da 0 ad 7 comprendono traffico per il quale la sorgente prevede una forma di controllo delle congestioni (rallentamento della trasmissione dei pacchetti). I restanti valori specificano tipi di traffico per cui non è previsto controllo della congestione (applicazioni "real time" tipo streaming etc. che richiedono un invio di pacchetti a ritmo costante). I valori più alti dovrebbero essere riservati solo ai pacchetti che il mittente ritiene essenziale non vengano persi.

Sicurezza con IPv6

IPv6 integra due meccanismi per provvedere servizi di trasmissione sicura a livello network. Tali meccanismi possono essere utilizzati congiuntamente o meno per soddisfare le diverse esigenze degli utenti.
Il primo meccanismo consente il controllo di identità ed integrità dei datagram senza però mascherarne il contenuto. Non è stato specificato un meccanismo per la generazione del contenuto dei campi di autenticazione, ma è stato suggerito l'utilizzo dei codici MD5. Questo tipo di controllo di sicurezza non

genera problemi legali nei paesi in cui l'impiego o l'esportazione delle tecniche di crittografia sono severamente regolamentati.

Il secondo meccanismo invece fornisce meccanismi, indipendenti da algoritmi specifici, per la crittografia ed il controllo di integrità dei pacchetti.

IPv6 in sintesi

IPv6 si propone di:

- supportare miliardi di host, anche con un meccanismo inefficiente di assegnazione degli indirizzi
- ridurre le dimensioni delle tabelle di routing, a seguito di una minore frammentazione dello spazio degli indirizzi
- diminuire il carico di lavoro dei router
- gestire la QoS
- offrire servizi di autenticazione e riservatezza
- lasciare un margine per l'evoluzione del protocollo
- coabitare, per il tempo necessario alla transizione, con i vecchi protocolli di rete.

La transizione verso IPv6

Fino a quando la connettività IPv6 non sarà largamente diffusa e supportata dall'infrastruttura di instradamento saranno necessari meccanismi per l'interoperabilità di reti IPv4 – IPv6. Si prevede una coabitazione prolungata per i due protocolli data l'enorme base installata di dispositivi di routing IPv4. La chiave del successo della diffusione di IPv6 sarà vincolata alla compatibilità con IPv4 dei meccanismi di transizione proposti ed alla loro semplicità di implementazione.
Tali meccanismi possono essere classificati in tre gruppi principali:

- Dual stack
- Tunneling
- Traduzione

Il meccanismo *dual stack* prevede la coabitazione degli stack IPv4 ed IPv6 su di ogni nodo della rete. Ogni nodo avrà per tanto indirizzi IPv4 ed IPv6. Si tratta di un metodo di semplice applicazione, anche grazie all'ormai diffuso supporto per IPv6 dei principali sistemi operativi. Tuttavia può essere necessario creare due routing table e prevedere due meccanismi di routing diversi per i due protocolli.

Il meccanismo di *tunneling* presenta un gran numero di varianti: prevede sostanzialmente di incapsulare pacchetti IPv6 entro pacchetti IPv4 in modo da connettere segmenti di rete IPv6 tra loro isolati.

La *traduzione* è richiesta quando un nodo che supporti esclusivamente IPv4 tenti di comunicare con un nodo esclusivamente IPv6.

I meccanismi di transizione sono discussi nello RFC 2893.

Il meccanismo di transizione Dual Stack

Un nodo "Dual stack" possiede entrambi i protocolli v4 e v6 ed è per tanto in grado di inviare e ricevere datagrammi di ogni tipo.

Tale nodo, indicato negli RFC come nodo IPv4/IPv6, possiede indirizzi di ambo i tipi e deve essere in grado di ottenere gli IPv4 attraverso assegnazione DHCP e gli IPv6 per mezzo dei meccanismi previsti dal protocollo (stateless address autoconfiguration).

I nodi IPv4/IPv6 devono anche essere provvisti di librerie per la risoluzione DNS che siano in grado di gestire sia le entries di tipo A, caratteristiche di IPv4, sia le entries di tipo A6 ed AAAA definite per gli indirizzi IPv6. Poiché la risoluzione dell'hostname potrebbe restituire indirizzi di entrambi i tipi, devono essere previsti meccanismi per la scelta del protocollo da impiegare. Può venire impostato un comportamento di default che preveda filtraggio ed ordinamento dei risultati, oppure il compito può venire demandato alle singole applicazioni. Qualsiasi implementazione delle librerie per la risoluzione DNS deve prevedere questa seconda opportunità.

E' una raccomandazione piuttosto superflua ricordare che non vale la pena fornire di entries DNS di tipo AAAA e A6 nodi dual stack che non siano connessi, fisicamente o per mezzo di tunneling, alla dorsale IPv6: il risultato, prevedibilmente, sarebbe di lunghe attese di time out, mentre gli host, privi di un percorso tra loro, cercano inutilmente di scambiarsi pacchetti IPv6 attraverso una rete IPv4.

Capitolo II
Il sistema ed il web server:
FreeBSD, Apache, PHP.

Scelta dei pacchetti software

Le ragioni che ci hanno spinto all'adozione di FreeBSD, Apache e PHP per lo sviluppo del nostro sito web sono principalmente tre:

- i pacchetti selezionati sono *open source*: questo, brevemente, significa gratuità e libertà di sperimentare con il loro codice. Caratteristiche attraenti in ambiente accademico, ma non solo;
- il sistema è notoriamente robusto e sicuro: la lista dei 50 server con uptime medio più lungo di Netcraft (http://uptime.netcraft.com/up/today/top.avg.html) è dominata da sistemi BSD che eseguono server Apache;
- per le due ragioni già menzionate, FreeBSD e Apache sono ampiamente diffusi sul mercato: ulteriore motivo per approfondirne la conoscenza!

Va aggiunto inoltre che FreeBSD è rinomato per la qualità dell'implementazione degli stack di rete, il che lo rende piattaforma d'elezione per gli scopi di questo studio.

FreeBSD

FreeBSD è un sistema operativo di tipo UNIX, derivato dal ramo di sviluppo di Berkeley, in grado di supportare numerose piattaforme hardware diverse. Diversamente da Linux, FreeBSD è interamente sviluppato da un unico gruppo. Kernel, utilità di sistema e device drivers fanno tutti parte di uno stesso albero CVS. Possiede una nutrita dotazione di software (chiamati ports, o nel caso di software precompilato, packages) ed offre un certo grado di compatibilità con il software sviluppato per Linux e disponibile solo in forma di eseguibile.
Sull'architettura di FreeBSD si basano, più o meno direttamente, molti altri sistemi operativi. tra questi vale senza dubbio la pena citare Mac OS X.

Apache

Apache è un server http multipiattaforma ed open source. Dall'aprile del 1996 è il server web più diffuso e nel novembre 2005 era in esecuzione sul 71% dei server web (dati Netcraft). Inizialmente sviluppato come evoluzione ed estensione del server NCSA, in occasione della release 2.0 è stato completamente riscritto. Apache è alla base dei cosiddetti pacchetti AMP (dalle iniziali di Apache, MySQL e PHP/Python/Perl): soluzioni integrate per lo sviluppo di siti web dal contenuto dinamico. Ad Apache spettano compiti di presentazione dell'informazione ai client, mentre MySQL gestisce l'immagazzinamento dell'informazione ed i linguaggi di scripting offrono i meccanismi di elaborazione server-side per la generazione di pagine dinamiche a partire dai contenuti del database. Essendo costituite di software open source, le soluzioni AMP sono assai diffuse e fanno comunemente parte delle distribuzioni Linux e BSD-based. Apache è anche distribuito come parte integrante di numerosi pacchetti commerciali tra cui Orale Database e Mac OS X.

Tra le funzioni offerte da Apache vale la pena citare aliasing e host virtuali, autenticazione, supporto per SSL e TLS, logging personalizzabile e, soprattutto, a partire dalla versione 2, il supporto per IPv6.

PHP

PHP è un linguaggio di scripting server-side interpretato di alto livello. Ciò significa che ogni qualvolta un client richieda un documento contenente codice PHP, quest'ultimo dovrà essere interpretato da un pre-processore che genererà codice HTML di conseguenza. Poiché l'elaborazione avviene sul lato server, ciò che verrà presentato al client sarà sempre e solo HTML puro, così da non richiedere altro ai browser che la capacità di formattarlo correttamente.

Il PHP nasce come una raccolta di script freeware in PERL, poi riscritti come binari CGI, nel 1994. Il “**P**ersonal **H**ome **P**age Tools” venne rilasciato nel 1995 quando, insieme agli script, venne distribuito anche un interprete. Nel 1997, a seguito di una riscrittura del parser, l’acronimo venne ridefinito “PHP: Hypertext Preprocessor”. A partire dalla versione 3, rilasciata nel 1998, il motore di PHP è stato ribattezzato Zend Engine.

PHP comprende un vasto numero di librerie per l’interazione con server di ogni tipo, dbms, archivi compressi, grafica, multimedia.

Dalla versione 3 in poi PHP offre funzioni di programmazione orientata agli oggetti.

Capitolo III

Firewalling

Sicurezza e reti.

La possibilità di connettere un numero arbitrario di computer tra loro è contemporaneamente una grande opportunità e fonte di problemi. Fin troppo in fretta ci si è abituati alla comodità di condividere quasi istantaneamente l'informazione. Questa nuova possibilità sta rapidamente mutando il modo di lavorare e di vivere di milioni di persone. Di tutti costoro, però, solo una piccola parte è conscia della quantità di pericoli che tutto ciò comporta. Sparsi su di un numero ignoto di calcolatori interconnessi attraverso le topologie di rete più disparate, risiedono grandi quantità di dati sensibili: informazioni personali, progetti industriali segreti, analisi riservate.

Si pongono problemi di sicurezza per i contenuti e per l'infrastruttura stessa: come prevenire le intrusioni? Come impedire il diffondersi di software pernicioso? Come fare tutto questo, tenendo presente che la maggior parte degli utenti non ha neppure la più vaga idea dei rischi che corre?

Firewalling

Un firewall, sia esso hardware oppure software, è uno strumento che permette di separare una sottorete, della quale si presume facciano parte solo macchine sicure, da tutto il resto, attraverso il controllo del traffico tra differenti *zones of trust.* Ad esempio, è possibile separare una intranet aziendale, ritenuta una zona affidabile, dal resto di internet, ritenuta altamente inaffidabile. Lo scopo finale è quello di controllare le connessioni da e verso l'interno di una sottorete per mezzo di una policy basata sul principio di minimo privilegio: ogni utente o processo deve poter accedere solo a ciò che è strettamente necessario.

Come già notato in precedenza, la topologia delle reti esistenti è molto varia. Perché una sottorete sia effettivamente protetta occorre che tutto il traffico in

ingresso od in uscita da essa possa essere “ispezionato”, non diversamente da quanto avveniva nelle antiche città circondate da mura, l’accesso alle quali era permesso solo attraverso porte presidiate da corpi di guardia. Il primo passo nella configurazione dei firewall consiste dunque nell’identificare i punti di passaggio obbligati del traffico nella sottorete. Ciò permette di rendere sicura la zona “presidiando” solo un numero ristretto di nodi.

Tipi di firewall

E’ possibile classificare diversi tipi di firewall a seconda del livello a cui operano:

- Network Layer firewall
- Application Layer firewall

Un firewall può o meno tenere traccia dello stato delle connessioni. In questo caso si distingue tra:

- Stateful firewall
- Stateless firewall

Network Layer firewall

I firewall operanti a livello network sono detti anche packet filters. In generale essi operano scartando tutti i pacchetti che non corrispondono ad un set di regole definite dal responsabile della sicurezza o dal fornitore del firewall, mentre instradano i restanti. Simili firewall sono spesso implementati nei router hardware e nei sistemi operativi che posseggano capacità di routing, quali FreeBSD, Linux, Windows Server.

Il filtraggio avviene leggendo gli header (e solo gli header) dei pacchetti, in base a parametri quali indirizzo IP o porta di provenienza e di destinazione del pacchetto, nome del dominio di provenienza nonché informazioni sui

protocolli di rete di livello superiore. Il processo di filtraggio in generale è realizzato in modo rapido e del tutto trasparente all'utente.

Application Layer firewall

Si tratta di firewall operanti a livello applicativo dello stack di rete, in grado di intercettare tutto il traffico da e per una determinata applicazione. Anziché limitarsi a scartare i pacchetti che non obbediscono alle regole imposte e ad instradare i restanti, un simile firewall si comporta come un server proxy per una quantità di applicazioni diverse. E' essenziale comprendere che questo tipo di firewall non compie operazioni di routing. Tutto il traffico viene fermato dal firewall: è poi quest'ultimo che, qualora lo giudichi accettabile, instaura una connessione appropriata.

Lavorando a livello applicativo è possibile effettuare controlli più approfonditi sul contenuto stesso dei pacchetti, quali prevenire la trasmissione di certi tipi di file oppure impedire lo sfruttamento di pecche logiche note del software in esecuzione sulle macchine protette senza per questo dover rinunciare ad utilizzarlo.

In linea di principio, l'ispezione dei contenuti di un pacchetto potrebbe anche prevenire la diffusione di virus ed altro software pernicioso. Tuttavia la complessità del problema ha finora scoraggiato un simile approccio.

Il firewalling a livello applicativo è divenuto possibile solo in tempi relativamente recenti, poiché il carico computazionale da esso richiesto è piuttosto elevato. Tuttavia i vantaggi rispetto al packet filtering sono evidenti: la capacità di analizzare il contenuto di una connessione http può prevenire ad esempio il tunneling di altri protocolli attraverso di esso, cosa altrimenti impossibile.

Stateless firewall

Gli stateless firewall operano senza tener conto del fatto che il pacchetto in analisi faccia o meno parte di una connessione. Questo comportamento è tipico dei firewall più vecchi.

Un esempio tipico di cosa questo significhi è offerto dal protocollo FTP che, per operare, necessità di aprire connessioni su porte casuali. Uno stateless firewall riceve un pacchetto indirizzato ad una porta alta, non associata esplicitamente ad alcun servizio noto e perciò lo scarta. Il firewall ha appena interrotto una connessione legittima, poiché non aveva modo di sapere che il pacchetto facesse parte di una legittima connessione FTP.

Stateful firewall

Gli stateful firewall tengono traccia dello stato di tutte le connessioni che li attraversano. Questi firewall sono programmati in modo da poter distinguere quali pacchetti sono appropriati allo stato di una determinata connessione. Solo ai pacchetti che corrispondono ad uno stato noto e legittimo di una data connessione è consentito passare, mentre gli altri vengono scartati.

Uno stateful firewall è in grado di conservare in memoria gli attributi essenziali di una connessione, dal suo inizio e fino al termine. Questi attributi, noti collettivamente come “stato della connessione”, comprendono in genere gli indirizzi delle macchine e le porte coinvolte nella comunicazione, nonché i numeri dei pacchetti che ne fanno parte. Dal punto di vista computazionale la maggior parte del carico è dovuto all’instaurazione delle connessioni; i successivi pacchetti vengono ispezionati rapidamente poiché è sufficiente confrontare il contenuto dei loro header con le voci presenti nella tabella degli stati delle connessioni mantenuta dal firewall. La chiusura di una connessione comporta l’eliminazione della voce relativa dalla tabella.

Il protocollo TCP instaura connessioni attraverso una procedura detta Three-Way Handshake. Quando un client inizia una nuova comunicazione, esso invia un pacchetto nella cui intestazione il bit SYN viene impostato ad 1. Uno stateful firewall considera ciascuno di questo pacchetti come una nuova connessione, cui assocerà lo stato *NEW*. Se il servizio richiesto è disponibile sulla macchina destinataria, questa risponderà settando i bit SYN ed ACK ad 1 nel pacchetto di risposta. Se il client risponde con un ulteriore ACK, la connessione entrerà nello stato *ESTABLISHED*. Il comportamento tipico di uno stateful firewall potrebbe ad esempio prevedere di lasciare uscire tutti i pacchetti da una intranet e di lasciare entrare solo pacchetti facenti parte di una connessione ESTABILISHED, consentendo un controllo del traffico più fine rispetto ad uno stateless firewall.. Le connessioni che rimangono inattive per un certo periodo di tempo vengono eliminate dalla tabella per prevenirne lo riempimento.

Il dover mantenere una tabella delle connessioni è al tempo stesso il punto di forza e la debolezza degli stateful firewall. Attualmente il più comune attacco di tipo Denial of Service è quello mirato al saturare le connessioni monitorate dai firewall.

Personal firewall

Una tipologia relativamente nuova di firewall è rappresentata dei personal firewall. Essi vengono eseguiti direttamente sul computer dell'utente, anziché su hardware dedicato, e solitamente proteggono esclusivamente le macchine sulle quali risiedono. Fanno uso di interfacce intuitive e possono essere istruiti, tramite menù pop-up, riguardo ai permessi di accesso da garantire alle applicazioni che facciano richiesta di accesso alla rete. Si tratta, sostanzialmente, di un tentativo di mettere gli utenti meno esperti in grado di

proteggere i propri sistemi e di diffondere una certa cultura della sicurezza presso chi faccia abitualmente uso di reti.

Sezione II

Sviluppo di un sito web per il Laboratorio Reti su stack multiprotocollo IPv4-IPv6

Capitolo IV
Configurazione di Apache

Introduzione alla configurazione di Apache

Scrivere una guida esauriente alla configurazione ed all'utilizzo di uno soltanto degli strumenti impiegati nella realizzazione del progetto è molto al di là degli scopi di questa tesi. In effetti, si stima che la documentazione relativa ai sistemi AMP ammonti ad oltre 6000 pagine di testo standard. Ci si limiterà per tanto ad illustrare soltanto alcuni aspetti dei software utilizzati che possano tornare utili a chi dovesse trovarsi nelle condizioni di svolgere un compito simile.

I port delle versioni 1.3 e 2 di Apache sono inclusi nella release 5.4 di FreeBSD.

Per configurare il server occorre modificare il file httpd.conf, normalmente sito in /usr/local/etc/apache2/httpd.conf

httpd.conf

Questo è il file di configurazione principale di Apache. Esso contiene comandi di configurazione chiamati directives, che delineano il comportamento del server.

Il documento è diviso in tre sezioni:

- Directives che controllano il modus operandi del processo server definendo il "global environment"
- Directives relative al comportamento del server principale o di default, cioè quello che gestisce tutte le richieste che non sono rivolte ad un virtual host. Qui sono contenute anche le directives che contengono i valori di default per i virtual hosts.

- Directives relative al comportamento dei virtual hosts, che consentono ad un unico processo server di soddisfare richieste inviate a differenti indirizzi IP / hostname.

La modifica del file presuppone il riavvio del server perché i cambiamenti abbiano effetto.

Prima sezione: il global environment

La directive ServerRoot permette di specificare la radice dell'albero di directory contenente i file di configurazione e di logging del server. I percorsi di eventuali file di configurazione e logging saranno considerati come relativi alla server root.

```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
ServerRoot "/usr/local"
```

Le directive Timeout, KeepAlive, MaxKeepAliveRequests, KeepAliveTimeout hanno un notevole impatto sulle prestazioni del server.

Timeout definisce l'intervallo da attendere prima di restituire un messaggio di errore, nel caso che la trasmissione di messaggi GET, POST, PUT, ACK risulti rallentata per qualche ragione. L'intervallo è espresso in secondi.

```
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300
```

KeepAlive permette di prolungare le sessioni http in modo da soddisfare più richieste attraverso una unica connessione TCP. Questo garantisce sensibili miglioramenti delle prestazioni durante il trasferimento di documenti quali pagine ricche di immagini, in seguito alla riduzione di overhead.

```
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
```

```
#
KeepAlive On
```

Le directive MaxKeepAliveRequests e KeepAliveTimeout permettono di raffinare ulteriormente il controllo delle connessioni, la prima regolando il numero di richieste che possono essere soddisfatte da un'unica connessione, la seconda il tempo d'attesa massimo tra due richieste prima che la connessione venga terminata. Bisogna tuttavia essere in grado di formulare ipotesi realistiche sul tipo di traffico che il server dovrà sostenere per ottimizzare questi valori: quanta grafica contengono le pagine? per quanto tempo insisterà mediamente un utente sulla stessa pagina? quanti utenti accederanno contemporaneamente al server? è possibile che si esauriscano le connessioni disponibili?

```
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 15
```

La direttiva Listen consente di configurare quali porte debba ascoltare il server ed eventualmente gli indirizzi delle macchine abilitate alla connessione. E' possibile specificare il protocollo da utilizzarsi su di una data porta, anche se normalmente ciò non è necessario: può tornare utile quando si desideri instaurare una connessione https su di una porta diversa da quella di default.

```
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
#
#Listen 12.34.56.78:80

Listen 80
```

Questa sezione contiene le istruzioni relative al caricamento di moduli all'avvio del server. La struttura modulare del server consente notevoli vantaggi in termini di flessibilità: è possibile abilitare o disabilitare diverse funzioni a run-time anziché in fase di compilazione ed aggiungere moduli di terze parti in qualsiasi momento dopo l'installazione. Il prezzo da pagare è una certa riduzione delle prestazioni sotto alcuni sistemi operativi. Si noti l'ultima riga del blocco, relativa al modulo per PHP.

```
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule access_module libexec/apache2/mod_access.so
LoadModule auth_module libexec/apache2/mod_auth.so
LoadModule auth_anon_module libexec/apache2/mod_auth_anon.so
LoadModule auth_dbm_module libexec/apache2/mod_auth_dbm.so
#LoadModule auth_digest_module libexec/apache2/mod_auth_digest.so
#LoadModule file_cache_module libexec/apache2/mod_file_cache.so
LoadModule charset_lite_module libexec/apache2/mod_charset_lite.so
#LoadModule cache_module libexec/apache2/mod_cache.so
#LoadModule disk_cache_module libexec/apache2/mod_disk_cache.so
LoadModule include_module libexec/apache2/mod_include.so
LoadModule deflate_module libexec/apache2/mod_deflate.so
LoadModule log_config_module libexec/apache2/mod_log_config.so
LoadModule logio_module libexec/apache2/mod_logio.so
LoadModule env_module libexec/apache2/mod_env.so
LoadModule mime_magic_module libexec/apache2/mod_mime_magic.so
LoadModule cern_meta_module libexec/apache2/mod_cern_meta.so
LoadModule expires_module libexec/apache2/mod_expires.so
LoadModule headers_module libexec/apache2/mod_headers.so
LoadModule usertrack_module libexec/apache2/mod_usertrack.so
LoadModule unique_id_module libexec/apache2/mod_unique_id.so
LoadModule setenvif_module libexec/apache2/mod_setenvif.so
<IfDefine SSL>
LoadModule ssl_module libexec/apache2/mod_ssl.so
</IfDefine>
LoadModule mime_module libexec/apache2/mod_mime.so
#LoadModule dav_module libexec/apache2/mod_dav.so
LoadModule status_module libexec/apache2/mod_status.so
LoadModule autoindex_module libexec/apache2/mod_autoindex.so
LoadModule asis_module libexec/apache2/mod_asis.so
LoadModule info_module libexec/apache2/mod_info.so
LoadModule cgi_module libexec/apache2/mod_cgi.so
#LoadModule dav_fs_module libexec/apache2/mod_dav_fs.so
LoadModule vhost_alias_module libexec/apache2/mod_vhost_alias.so
LoadModule negotiation_module libexec/apache2/mod_negotiation.so
LoadModule dir_module libexec/apache2/mod_dir.so
LoadModule imap_module libexec/apache2/mod_imap.so
```

```
LoadModule actions_module libexec/apache2/mod_actions.so
LoadModule speling_module libexec/apache2/mod_speling.so
LoadModule userdir_module libexec/apache2/mod_userdir.so
LoadModule alias_module libexec/apache2/mod_alias.so
LoadModule rewrite_module libexec/apache2/mod_rewrite.so
LoadModule php4_module        libexec/apache2/libphp4.so
```

Seconda sezione: il main server

La seconda sezione contiene directive configuranti il comportamento del main server, che risponde a tutte le richieste non rivolte ad un virtual host. Essa definisce anche il comportamento di default per i virtual host: perché un virtual host adotti un comportamento diverso, la relativa directive dovrà essere sovrascritta nella sezione appropriata.

Tutte le directive di questa sezione possono apparire anche nella sezione relativa ai virtual host.

La directive ServerAdmin contiene l'indirizzo e-mail dell'amministratore del server. L'indirizzo appare, ad esempio, nelle pagine d'errore generate automaticamente dal server. Può essere saggio utilizzare un account di posta riservato alla sola gestione del server.

```
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.  e.g. admin@your-domain.com
#
ServerAdmin you@example.com
```

ServerName viene utilizzato per specificare il nome e la porta che il server usa per identificarsi. E' importante che esso contenga una entry DNS valida per il server, od in alternativa il suo indirizzo IP affinchè eventuali redirection funzionino correttamente. Ad ogni modo il server dovrebbe essere in grado di determinare automaticamente questo valore al momento dell'avvio.

```
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If this is not set to valid DNS name for your host, server-generated
# redirections will not work.  See also the UseCanonicalName directive.
#
```

```
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#
#ServerName www.example.com:80
```

Diversamente dalla directive ServerRoot, questa specifica dove siano contenuti i documenti.

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/www/data/html"
```

Segue un certo numero di directive che regola l'accesso alle directory di un host. Per default viene stabilita una politica d'accesso generale molto restrittiva, e viene lasciato all'amministratore il compito di specificare comportamenti differenti che rispondano alle proprie esigenze.

```
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    AllowOverride None
    Order Deny,Allow
    Deny from all
</Directory>
```

La seguente linea specifica che i blocchi successivi sono da considerarsi applicati alla DocumentRoot.

```
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr/local/www">
```

La directive Options consente di specificare quali operazioni sono accettabili all'interno della directory. In questo caso è stato consentito di vedere il contenuto della directory come lista formattata qualora non sia prevista una pagina di indice. E' stato anche consentito l'utilizzo di link simbolici all'interno della directory.

```
# Possible values for the Options directive are "None", "All",
```

```
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs-2.0/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks
```

I file .htaccess permettono di sovrascrivere i permessi di accesso alle directory specificati in httpd.conf. La directive AllowOverride governa se e come la sovrascrittura sia possibile. Poiché l'utilizzo di .htaccess può causare problemi di sicurezza, se ne raccomanda l'utilizzo solo ad utenti esperti. Altrimenti si lasci questa directive impostata su None.

```
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None
```

Questa linea delimita il blocco di directive da applicare alla DocumentRoot.

```
</Directory>
```

DirectoryIndex specifica il nome del file che Apache restituirà qualora venga richiesto acceso ad una directory anziché ad un documento.

```
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents.  The MultiViews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.html.var
```

AccessFileName specifica il nome del file contenente informazioni aggiuntive di accesso da cercare in ogni directory. Come già detto, non è opportuno fare ricorso agli AccessFile se non si è esperti nell'uso del web server.

```
# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
#
AccessFileName .htaccess
```

ErrorLog specifica dove salvare i log dei messaggi di errore del server. E' possibile utilizzare un log file diverso per ogni host virtuale. Vale la pena ricordare che Apache 2 consente di personalizzare i log e che esistono numerose utility per l'analisi dei log. I log sono strumento d'elezione per la diagnosi dei problemi e per il monitoraggio della sicurezza e del corretto funzionamento di un server. Altre directive relative al logging di interesse sono LogLevel, LogFormat, CustomLog

```
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog /var/log/httpd-error.log
```

Alias consente di associare ad un percorso assoluto un nome "falso", attraverso il quale il contenuto della directory sia facilmente accessibile attraverso il web server. Nel caso illustrato il contenuto di "/usr/local/www/icons" è accessibile come "hostname/icons/".

```
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL.  So "/icons" isn't aliased in this
# example, only "/icons/".  If the fakename is slash-terminated, then the
# realname must also be slash terminated, and if the fakename omits the
# trailing slash, the realname must also omit it.
#
# We include the /icons/ alias for FancyIndexed directory listings.  If you
# do not use FancyIndexing, you may comment this out.
#
Alias /icons/ "/usr/local/www/icons/"

<Directory "/usr/local/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

ScriptAlias definisce le directory che contengono script che il server deve eseguire piuttosto che documenti da inviare ai client.

```
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
```

```
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/usr/local/www/cgi-bin/"

<IfModule mod_cgid.c>
#
# Additional to mod_cgid.c settings, mod_cgid has Scriptsock <path>
# for setting UNIX socket for communicating with cgid.
#
#Scriptsock            /var/run/cgisock
</IfModule>

#
# "/usr/local/www/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/usr/local/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

La funzione di redirect permette di informare i client sullo stato di documenti che in passato risiedevano nel namespace del server ma ora non più. Ciò consente in genere di redirigere i client al documento cercato.

```
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Example:
# Redirect permanent /foo http://www.example.com/bar
```

Le seguenti directive permettono di personalizzare l’output degli index generati automaticamente dal server in assenza di un file index.html. E’ possibile aggiungere icone specifiche per i differenti tipi di file e relativi commenti descrittivi.

```
#
# Directives controlling the display of server-generated directory listings.
#

#
# IndexOptions: Controls the appearance of server-generated directory
# listings.
#
IndexOptions FancyIndexing VersionSort

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions.  These are only displayed for
# FancyIndexed directories.
#
```

```
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes.  These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz
```

E’ possibile configurare il web server perché, attraverso la content negotiation, esso restituisca documenti nella lingua appropriata.

Tra le directive importanti occorre segnalare:

- DefaultLanguage che specifica la lingua dei documenti che non posseggono un tag specifico per la lingua. E’ bene utilizzare questa directive con cautela, poiché è preferibile non specificare alcuna lingua, piuttosto che una lingua sbagliata.

- AddLanguage consente di specificare quali tag usare per identificare le lingue
- LanguagePriority e ForceLanguagePriority vengono utilizzati nel corso della content negotiation per determinare quali lingue siano preferibili per il client.

```
# DefaultLanguage and AddLanguage allows you to specify the language of
# a document. You can then use content negotiation to give a browser a
# file in a language the user can understand.
#
# Specify a default language. This means that all data
# going out without a specific language tag (see below) will
# be marked with this one. You probably do NOT want to set
# this unless you are sure it is correct for all cases.
#
# * It is generally better to not mark a page as
# * being a certain language than marking it with the wrong
# * language!
#
# DefaultLanguage nl
#
# Note 1: The suffix does not have to be the same as the language
# keyword --- those with documents in Polish (whose net-standard
# language code is pl) may wish to use "AddLanguage pl .po" to
# avoid the ambiguity with the common suffix for perl scripts.
#
# Note 2: The example entries below illustrate that in some cases
# the two character 'Language' abbreviation is not identical to
# the two character 'Country' code for its country,
# E.g. 'Danmark/dk' versus 'Danish/da'.
#
# Note 3: In the case of 'ltz' we violate the RFC by using a three char
# specifier. There is 'work in progress' to fix this and get
# the reference data for rfc1766 cleaned up.
#
# Catalan (ca) - Croatian (hr) - Czech (cs) - Danish (da) - Dutch (nl)
# English (en) - Esperanto (eo) - Estonian (et) - French (fr) - German (de)
# Greek-Modern (el) - Hebrew (he) - Italian (it) - Japanese (ja)
# Korean (ko) - Luxembourgeois* (ltz) - Norwegian Nynorsk (nn)
# Norwegian (no) - Polish (pl) - Portugese (pt)
# Brazilian Portuguese (pt-BR) - Russian (ru) - Swedish (sv)
# Simplified Chinese (zh-CN) - Spanish (es) - Traditional Chinese (zh-TW)
#
AddLanguage ca .ca
AddLanguage cs .cz .cs
AddLanguage da .dk
AddLanguage de .de
AddLanguage el .el
AddLanguage en .en
AddLanguage eo .eo
AddLanguage es .es
AddLanguage et .et
AddLanguage fr .fr
AddLanguage he .he
AddLanguage hr .hr
AddLanguage it .it
AddLanguage ja .ja
```

```
AddLanguage ko .ko
AddLanguage ltz .ltz
AddLanguage nl .nl
AddLanguage nn .nn
AddLanguage no .no
AddLanguage pl .po
AddLanguage pt .pt
AddLanguage pt-BR .pt-br
AddLanguage ru .ru
AddLanguage sv .sv
AddLanguage zh-CN .zh-cn
AddLanguage zh-TW .zh-tw

#
# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
#
# Just list the languages in decreasing order of preference. We have
# more or less alphabetized them here. You probably want to change this.
#
LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl nn no pl pt
pt-BR ru sv zh-CN zh-TW

#
# ForceLanguagePriority allows you to serve a result page rather than
# MULTIPLE CHOICES (Prefer) [in case of a tie] or NOT ACCEPTABLE (Fallback)
# [in case no accepted languages matched the available variants]
#
ForceLanguagePriority Prefer Fallback
```

Similmente a quanto visto per le lingue, è possibile controllare i set di caratteri utilizzati per la visualizzazione dei documenti.

```
# Commonly used filename extensions to character sets. You probably
# want to avoid clashes with the language extensions, unless you
# are good at carefully testing your setup after each change.
# See http://www.iana.org/assignments/character-sets for the
# official list of charset names and their respective RFCs.
#
AddCharset ISO-8859-1  .iso8859-1  .latin1
AddCharset ISO-8859-2  .iso8859-2  .latin2 .cen
AddCharset ISO-8859-3  .iso8859-3  .latin3
AddCharset ISO-8859-4  .iso8859-4  .latin4
AddCharset ISO-8859-5  .iso8859-5  .latin5 .cyr .iso-ru
AddCharset ISO-8859-6  .iso8859-6  .latin6 .arb
AddCharset ISO-8859-7  .iso8859-7  .latin7 .grk
AddCharset ISO-8859-8  .iso8859-8  .latin8 .heb
AddCharset ISO-8859-9  .iso8859-9  .latin9 .trk
AddCharset ISO-2022-JP .iso2022-jp .jis
AddCharset ISO-2022-KR .iso2022-kr .kis
AddCharset ISO-2022-CN .iso2022-cn .cis
AddCharset Big5        .Big5       .big5
# For russian, more than one charset is used (depends on client, mostly):
AddCharset WINDOWS-1251 .cp-1251   .win-1251
AddCharset CP866       .cp866
AddCharset KOI8-r      .koi8-r .koi8-ru
AddCharset KOI8-ru     .koi8-uk .ua
AddCharset ISO-10646-UCS-2 .ucs2
AddCharset ISO-10646-UCS-4 .ucs4
AddCharset UTF-8       .utf8
```

```
# The set below does not map to a specific (iso) standard
# but works on a fairly wide range of browsers. Note that
# capitalization actually matters (it should not, but it
# does for some browsers).
#
# See http://www.iana.org/assignments/character-sets
# for a list of sorts. But browsers support few.
#
AddCharset GB2312      .gb2312 .gb
AddCharset utf-7       .utf7
AddCharset utf-8       .utf8
AddCharset big5        .big5 .b5
AddCharset EUC-TW      .euc-tw
AddCharset EUC-JP      .euc-jp
AddCharset EUC-KR      .euc-kr
AddCharset shift_jis   .sjis
```

AddTypes permette di gestire file appartenenti a specifici tipi MIME sovrascrivendo le impostazioni di default. Da notare come siano stati inseriti anche i file .html tra i tipi MIME che devono essere pre-processati dall'engine PHP. In questo modo verrà effettuato il parsing di ogni singola pagina web, indipendentemente dal fatto che contenga codice PHP, ma la tecnologia di scripting utilizzata dal sito rimarrà nascosta al pubblico. AddEncoding e AddHandlers svolgono compiti affini, fornendo decompressione "al volo" di contenuti e la possibilità di effettuare operazioni sui file indipendentemente dal loro tipo.

```
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
#AddType application/x-tar .tgz
AddType application/x-httpd-php .php .phtml .html
AddType application/x-httpd-php-source .phps
#
# AddEncoding allows you to have certain browsers uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above.
#
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz
#
# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

#
# AddHandler allows you to map certain file extensions to "handlers":
# actions unrelated to filetype. These can be either built into the server
```

```
# or added with the Action directive (see below)
#
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
#
#AddHandler cgi-script .cgi
```

Terza sezione: host virtuali

E’ possibile associare ad una medesima macchina più domini/hostname per mezzo dell’uso di virtual host. Il server Apache gestirà in modo appropriato e personalizzato le richieste ai diversi host virtuali configurati. Questo consente, tra le altre cose, di lavorare in modo semplice allo sviluppo di un sito web: è sufficiente mantenere sulla medesima macchina due o più copie del sito, in differente stato di sviluppo. Ad esempio:

- www.sitename.net potrà essere accessibile al pubblico
- beta.sitename.net potrà essere accessibile con password solo ai tester
- devel.sitename.net potrà avere un accesso ancora più ristretto di beta; solo da macchine che abbiano un indirizzo IP appartenente ad un pool ristretto.

Intervenendo su httpd.conf è sufficiente rinominare le sezioni relative ai nomi dei virtual host e modificare i permessi di accesso perché un sito passi da devel a beta e poi a www.

Capitolo V

Configurazione del firewall

Scelta del firewall da utilizzare

FreeBSD viene distribuito con tre differenti pacchetti firewall, per venire incontro alle esigenze di differenti tipologie di utenti:

- IPFILTER (IPF)
- IPFIREWALL (IPFW)
- OpenBSD's Packet Filter (PF)

I tre pacchetti utilizzano una diversa sintassi per le regole di inclusione del traffico e meccanismi differenti per il controllo dello stato delle connessioni; possono essere usati in congiunzione con pacchetti per la modellazione del traffico (funzioni di Quality of Service) e sono in grado di svolgere compiti di Network Address Translation.

Vista la notevole quantità di documentazione disponibile, la scelta è caduta su PF.

Abilitazione di PF

PF è disponibile come modulo per il kernel caricabile dinamicamente. Perché il modulo venga caricato occorre inserire la seguente linea in /etc/rc.conf :

```
pf_enable="YES"
```

Conviene anche abilitare le seguenti opzioni all'interno di /etc/rc.conf :

```
pf_rules="/etc/pf.conf"
```

per specificare il percorso del file contenente le regole di inclusione del traffico;

```
pflog_enable="YES"              # start pflogd(8)
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
```

per attivare le funzioni di logging del firewall e specificare il percorso del file di log.

Impostazione delle regole per il firewall

Il file di configurazione di PF si compone di sette sezioni:

- macros
- tables
- options
- scrub
- queueing
- translation
- filtering

Le sezioni devono essere disposte necessariamente nell'ordine mostrato.

Macros contiene variabili, definite dall'utente, che rappresentano specifiche interfacce di rete od indirizzi IP.

Tables contiene variabili che corrispondono a liste di interfacce od indirizzi IP.

Options permette di impostare il comportamento di default di PF: ad esempio, la linea

```
set block-policy option
```

stabilisce cosa debba essere fatto dei pacchetti bloccati dal firewall. E' naturalmente possibile specificare comportamenti diversi dal default nella definizione delle regole all'interno della sezione filtering.

Scrub concerne la *normalizzazione* dei pacchetti. Il processo di normalizzazione permette di eliminare alcune situazioni ambigue che possono essere sfruttate da malintenzionati per violare il sistema. Ad esempio, un pacchetto frammentato potrebbe non essere riconosciuto dai sistemi di identificazione dei tentativi di intrusione (network intrusion detections systems, NIDS) come contenente una minaccia, se esso viene riassemblato completamente soltanto sullo host destinatario. Senza una conoscenza precisa dei meccanismi di riassemblamento sugli end point i NIDS non possono fare previsioni sul modo in cui una sequenza di pacchetti frammentati verrà riassemblata e trattata. L'ignoranza della topologia della rete tra NIDS ed end system rende impossibile prevedere se un pacchetto con basso TTL raggiungerà la sua destinazione: il NIDS potrebbe quindi formarsi un modello erroneo dello stato delle connessioni.

Queueing consente di governare la priorità del traffico in uscita, ad esempio garantendo latenza inferiore alle applicazioni real time.

Translation controlla il NAT e l'inoltro dei pacchetti.

Filter Rules contiene le regole in base alle quali il traffico verrà effettivamente filtrato.

Passiamo ora all'analisi del file di configurazione:
ext_if ed int_if identificano rispettivamente l'interfaccia esterna, collegata ad internet, e l'interfaccia interna, collegata al laboratorio di reti, del firewall.

```
# Macros:.
ext_if="rl0"
int_if="tl0"
```

webserver è una tabella contenente gli indirizzi IPv4 ed IPv6 del server web del laboratorio.

```
# Tables:
table <webserver> { 193.206.71.158, 2001:760:2000:1000::ffff }
```

Nelle opzioni il firewall viene impostato in modo da scartare i pacchetti che non superano il filtro, senza inviare alcuna risposta a chi ha generato il pacchetto scartato.

```
# Options:
set block-policy drop
```

La seguente linea impone la normalizzazione di tutti i pacchetti. Il processo aumenta considerevolmente la sicurezza del sistema, al prezzo della possibile perdita di alcuni pacchetti dalla struttura anomala ancorché legittima.

```
# Normalization:
scrub in all
```

Le sezioni relative a queueing e NAT non sono necessarie nel caso in esame, dunque si passa direttamente alle regole per il filtraggio del traffico.

```
# Filtering:
```

La prima riga blocca l'intero traffico in ingresso a tutte le interfacce. E' una scelta molto prudente che, congiuntamente alle regole che seguiranno, permetterà l'ingresso solo al traffico specificato.

```
block in all
```

Le due regole seguenti sono piuttosto importanti: regolano il traffico attraverso il tunnel IPv4-IPv6 del laboratorio reti.

```
pass in on rl0 inet proto ipv6 from 193.204.34.254 to rl0 keep state
pass out on rl0 inet proto ipv6 from rl0 to 193.204.34.254 keep state
```

Vale la pena studiarle attentamente anche per comprendere meglio la struttura di una regola di filtraggio:

`pass in` e `pass out` specificano che queste regole permetteranno il passaggio in ingresso ed in uscita del traffico. Poiché in precedenza abbiamo bloccato tutto il traffico, ora occorre stabilire selettivamente ciò che può passare.

`on rl0` informa che le regole andranno applicate all'interfaccia rl0. rl0 è anche definita ext_if, interfaccia esterna, nella sezione macros. Sarebbe stato possibile utilizzare il nome della macro anziché il nome dell'interfaccia. Dunque questo filtro deve essere applicato a livello dell'interfaccia esterna.

`inet` specifica che il traffico deve essere di tipo IPv4. Come detto in precedenza infatti, il traffico IPv6 attraverso un tunnel viene incapsulato in IPv4.

`proto ipv6` specifica che solo il traffico IPv4 che incapsuli pacchetti IPv6 è da ritenersi interessato alla regola.

Il significato di `from 193.204.34.254 to rl0` e `from rl0 to 193.204.34.254` è piuttosto intuitivo: specificano quali siano le sorgenti ed i destinatari legittimi dei due flussi di traffico individuati dalle regole.

In fine `keep state` informa il firewall che si vuole monitorare lo stato delle connessioni per il traffico interessato da questa regola.

Per ragioni di sicurezza si preferisce non commentare il resto del file di configurazione del firewall in questa sede.

Capitolo VI

Il sito web

Scopo del sito

E' stata richiesta la realizzazione di un sito che funga da vetrina per il laboratorio reti. In particolare si desiderava uno spazio attraverso cui poter

- fornire agli studenti materiale didattico relativo alle esercitazioni
- tenere traccia delle attività di ricerca effettuate dal laboratorio
- pubblicizzare l'adesione della struttura ai progetti per la migrazione verso IPv6.

Poiché il docente responsabile del laboratorio possiede già un proprio sito attraverso cui comunicare con gli studenti, non è stata ritenuta necessaria la realizzazione di una bacheca per gli avvisi.

Data l'assenza di particolari necessità di interattività da parte degli utenti, si è scelto di non fare ricorso a CMS od a DBMS per generare contenuto dinamico. Si è scelto invece di utilizzare PHP per sottolineare come il sito sia ospitato su di un server Dual Stack.

Analisi del template per le pagine del sito.

E' stato spesso sottolineato come l'uniformità della grafica delle pagine di un medesimo sito sia percepita benevolmente dai visitatori. Si è per tanto optato di utilizzare la tecnologia dei fogli di stile per generare un unico template applicabile a tutte le pagine del sito con uno sforzo minimo.

La pagina è costituita da una *intestazione* di dimensioni e contenuti fissi, da un menù di navigazione, sito sulla destra, e da un'area, di dimensioni variabili, per la visualizzazione dei contenuti.

Figura 2 - Layout del template per le pagine del sito.

Poiché si ritiene che i lettori abbiano una certa dimestichezza con le tecnologie HTML e CSS, verranno qui discusse solo alcune scelte di progettazione.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w2.org/TR/REC-html40/strict.dtd">


<HTML>


<!-- Inizio sezione HEAD -->
<HEAD>

      <TITLE>Laboratorio reti, Universita' di Pavia, Sede di Mantova</TITLE>
```

Si è fatto uso di uno script in PHP per determinare il protocollo utilizzato dai client per la connessione al server web. Lo script originale, realizzato da Alexander Merz e rilasciato sotto licenza PHP 2.0, è stato modificato da Dario Genova ed Emanuele Goldoni affinchè la pagina web mostrata dal browser includa un foglio di stile basato su di un tema di colore verde, nel caso la connessione sia di tipo IPv6, rosso se di tipo IPv4.

```
      <LINK rel=stylesheet href="<?php include"IPvChecker.php" ?>"
type="text/css">


</HEAD>


<!-- Inizio sezione BODY -->
<BODY>
```

La sezione BODY inizia con l'inclusione di un file detto "header". Tale file contiene codice HTML per la generazione dell'intestazione della pagina. Poiché l'intestazione è identica per tutte le pagine, si è scelto di mantenerne il codice separato dal resto. Modificando il solo file "header", i cambiamenti verranno applicati a tutte le pagine generate con questo template.

```
      <?php include"header" ?>
```

La "textnavarea" raggruppa l'area dei contenuti ed il menù di navigazione per semplificare il posizionamento sullo schermo degli elementi della pagina. Lo studio del CSS aiuterà a comprendere questa scelta.

```
      <DIV class="textnavarea">
```

Anche per la sezione del menu di navigazione si è fatto ricorso ad un include, per le ragioni già menzionate.

```
            <?php include"navmenu" ?>
```

Per concludere, nella sezione "textarea" vanno immessi i contenuti della pagina: nell'esempio sottostante il codice è quello della pagina iniziale.

```
            <DIV class="textarea">
```

Le “sectiontitlearea” sono utilizzate per definire i titoli delle sezioni testo.

```
                <DIV class="sectiontitlearea">
                      Home
                </DIV>
                <BR>
                UNIVERSITA' DEGLI STUDI DI PAVIA  -  Facolta' di Ingegneria
                      <img align="right" src="./pics/logo.jpg"><BR>
                Sede distaccata di Mantova (ex Convento S. Francesco)<BR>
                Laboratorio Reti (Polo Informatico)<BR>
                Via Scarsellini, 2 - 46100 Mantova<BR>
                Tel. (+39) - 0376 - 286.238<BR>
          </DIV>

      </DIV>

</BODY>
```

Per concludere, il codice html di “navmenu” ed “header”., le cui modifiche si riflettono su tutte le pagine.

```
<DIV class="navmenu">
      <DIV class="sectiontitlearea">
            Sezioni
      </DIV>
      <hr>
      <p><a href="index.html">Home</A></p>
      <hr>
      <p><a href="presentazione.html">Presentazione</a><br>
      <a href="didattica.html">Didattica</a><br>
      <a href="progettietesi.html">Progetti e Tesi</a><br>
      <hr>
      <img src="./pics/logosmall.jpg">
</div>


<DIV class="titlearea" align="center">
      <IMG src="./pics/labreti.jpg"><BR>
      <DIV class="sectiontitlearea">
            Universita' degli studi di Pavia, sede distaccata di Mantova.<BR>
            Responsabile: Prof. Giuseppe F. Rossi.
```

```
	</DIV>
</DIV>
```

Per quanto riguarda il foglio di stile, le sezioni principali sono tre. La prima definisce le impostazioni di default del body del documento html.

```
BODY
{
	background-image: URL("./pics/bgg.gif");
	font: 12pt/12pt sans-serif;
	color: #000000;
	background-color: #f0f0f0;
}
```

La seconda sezione riguarda la distribuzione delle aree di testo sullo schermo: titlearea rappresenta l'intestazione della pagina. E' stato scelto il posizionamento relativo per titlearea e per textnavarea poiché è il modo più semplice di posizionare sulla pagina elementi le cui dimensioni non sono note a priori.

```
DIV.titlearea
{
	border: thin solid;
	position: relative;
	margin: 1% 0% 1% 0%;
	padding: 1% 1% 1% 1%;
	background-color: #ffffff;
	min-width: 960px;
}

DIV.textnavarea
{
	position: relative;
}

DIV.sectiontitlearea
{
	position: relative;
	background-color: #009000;
	color: #ffffff;
	padding: 1% 1% 1% 1%;
}
```

navmenu e textarea invece hanno posizionamento assoluto, poiché si vuole che occupino porzioni ben definite dello schermo: poco meno del 20 % della larghezza per il menù di navigazione, poco meno dell'80% per l'area di testo.

```
DIV.navmenu
{
	border: thin solid;
	position: absolute;
	right: 0px;
	width: 17%;
	margin: 1% 0% 1% 1%;
	padding: 1% 1% 1% 1%;
	font: 12pt/18pt sans-serif;
	background-color: #ffffff;
	min-width: 154px;
}

DIV.textarea
{
	border: thin solid;
	position: absolute;
	margin: 1% 1% 1% 0%;
	padding: 1% 1% 1% 1%;
	width: 77%;
	background-color: #ffffff;
	min-width: 298px;
}
```

Da ultimo, la terza sezione delinea lo stile dei link sulla pagina.

```
A:link
{
	text-decoration:none;
}

A:visited
{
	text-decoration:none;
}

A:hover, A:active
{
	text-decoration: underline;
}
```

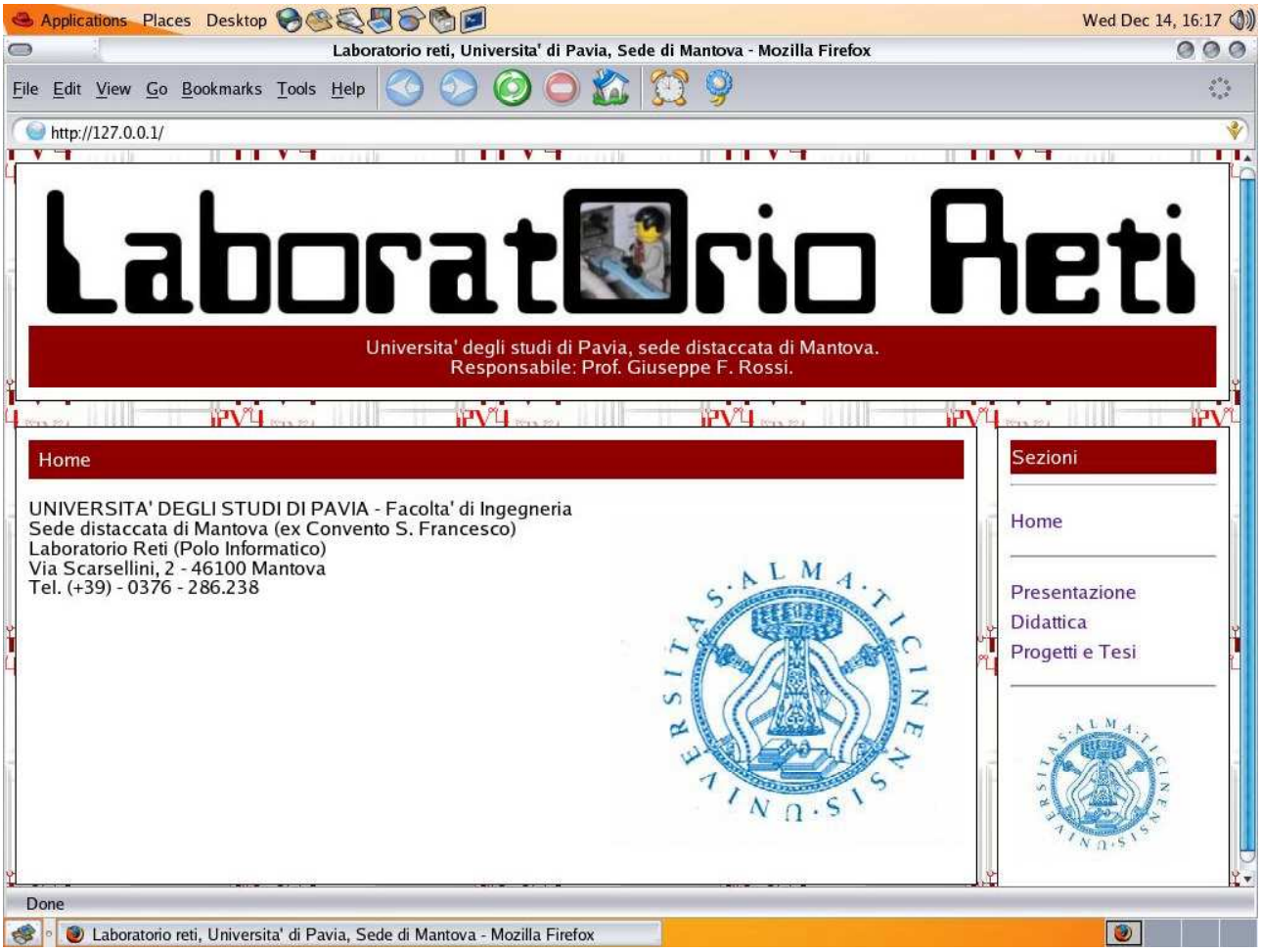

Figura 3 - Aspetto della home page del laboratorio, connessione effettuata utilizzando IPv4

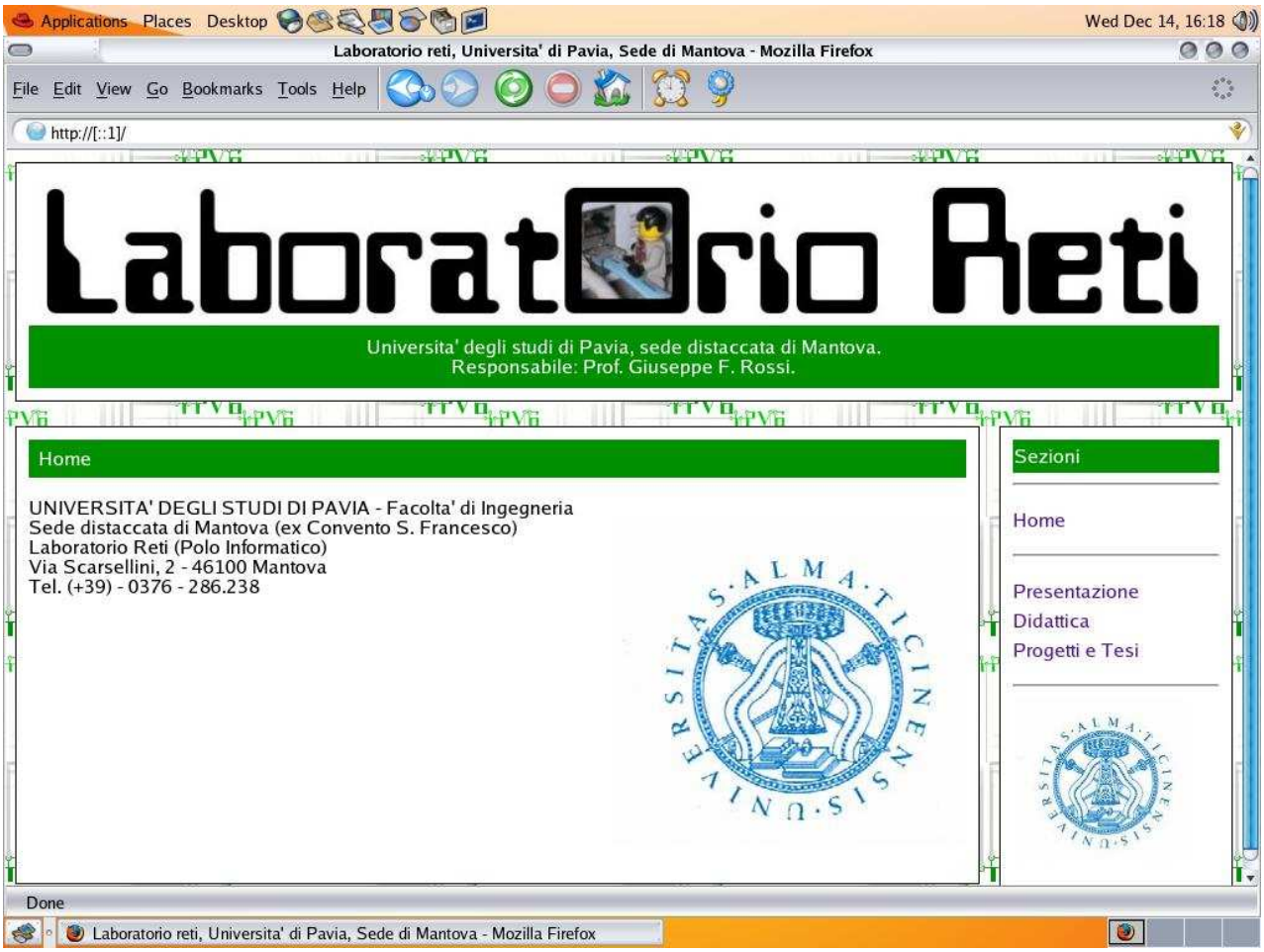

Figura 4 - Aspetto della home page del laboratorio, connessione effettuata utilizzando IPv6

Sezione III
Conclusioni

La tecnologia IPv6 appare ormai matura: tutto il software utilizzato nel corso di questo progetto dispone di librerie native per l'utilizzo del nuovo protocollo. L'attivazione del doppio stack su di una macchina FreeBSD non richiede che la modifica di poche righe in un file di configurazione. Apache 2 supporta IPv6 in modo completamente trasparente all'utente. La stesura di regole per il firewalling è sostanzialmente identica per i due protocolli.

Ma la versione di Apache maggiormente diffusa attualmente è ancora la 1.3. La bontà del software scritto per IPv4 e la flessibilità del vecchio protocollo, forse sottovalutata, sconsigliano i proprietari delle infrastrutture dall'operare la transizione.

L'introduzione di tecnologie quali NAT e CIDR ha momentaneamente ridotto l'urgenza di adottare un nuovo protocollo in grado di superare il problema dell'esaurimento dello spazio degli indirizzi. Similmente è possibile fornire almeno una parte dei servizi offerti da IPv6 tramite soluzioni che, per quanto arzigogolate, non richiedono massicce riconversioni di infrastrutture. Siamo ancora lontani dal dover allocare un indirizzo IP ad ogni elettrodomestico. Ma l'appuntamento con IPv6 è solo rimandato: i paesi del lontano oriente, Cina, Giappone, India, ed in minor misura l'Europa, storicamente penalizzati dalla distribuzione asimmetrica degli indirizzi IPv4, avvertono maggiormente l'urgenza della transizione ed i benefici maturati dalla partecipazione allo sviluppo del nuovo standard.

Bibliografia

IPv6

Internetworking con TCP/IP, Volume I, Quarta Edizione, D. Comer, Addison-Wesley

Reti di calcolatori, Quarta Edizione, A. S. Tanenbaum, Prentice Hall

IP Next generation overview, R. M. Hinden,
http://playground.sun.com/pub/ipng/html/INET-IPng-Paper.html

Wikipedia, http://en.wikipedia.org

RFC:
1924, 2374, 2460, 2461, 2462, 2463, 2464, 2893, 3513,

Firewall

Internetworking con TCP/IP, Volume I, Quarta Edizione, D. Comer, Addison-Wesley

Reti di calcolatori, Quarta Edizione, A. S. Tanenbaum, Prentice Hall

FreeBSD Handbook,
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html

PF: the OpenBSD Packet Filter
http://www.openbsd.org/faq/pf/

Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, M.Handley, V.Paxson, C. Kreibich
http://www.icir.org/vern/papers/norm-usenix-sec-01-html/index.html

Wikipedia, http://en.wikipedia.org

FreeBSD

FreeBSD Handbook,
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html

Apache

Apache http server documentation
http://httpd.apache.org/docs/2.2/

PHP5, Apache e MySQL, E. Naramore, J. Gerner, Y. Le Scouarnec, J. Stoltz, M. K. Glass, Hoepli Informatica

PHP

PHP: Documentation
http://www.php.net/docs.php

PHP5, Apache e MySQL, E. Naramore, J. Gerner, Y. Le Scouarnec, J. Stoltz, M. K. Glass, Hoepli Informatica

HTML

W3 Schools on line web tutorials
http://www.w3schools.com/

HTML, M. E. Holzschlag, Mondadori Informatica

CSS

W3 Schools on line web tutorials

http://www.w3schools.com/

HTML, M. E. Holzschlag, Mondadori Informatica

CSS Zen Garden

http://www.csszengarden.com/