Gestione della posta elettronica su Linux
Introduzione

Il sistema di posta elettronica di *UNIX* è formato dall'interazione di più componenti *software* che devono essere opportunamente configurati (sebbene le operazioni più complesse vengono svolte da un unico programma, il *Mail Transport Agent*). Nella trattazione faremo riferimento a **Sendmail**, il *Mail Transport Agent* più diffuso in ambiente *UNIX*. Data la complessità dell'argomento, si consiglia di fare riferimento al *Linux Documentation Project* (www.tldp.org) e ai testi di base sul sistema operativo *Linux*.

La gestione della posta elettronica in Unix

In questa unità didattica viene descritto il sistema di posta elettronica di *UNIX*. Il sistema deriva dall'interazione di più componenti *software* che devono essere opportunamente configurate (sebbene le operazioni più complesse vengono svolte da un unico programma, il *Mail Transport Agent*). Nella trattazione faremo riferimento a *Sendmail*, il *Mail Transport Agent* più diffuso in ambiente *UNIX*. Data la complessità dell'argomento, si consiglia di fare riferimento ai testi consigliati nella pagina di Risorse.

Generalità sulla posta elettronica in *Unix/Linux*

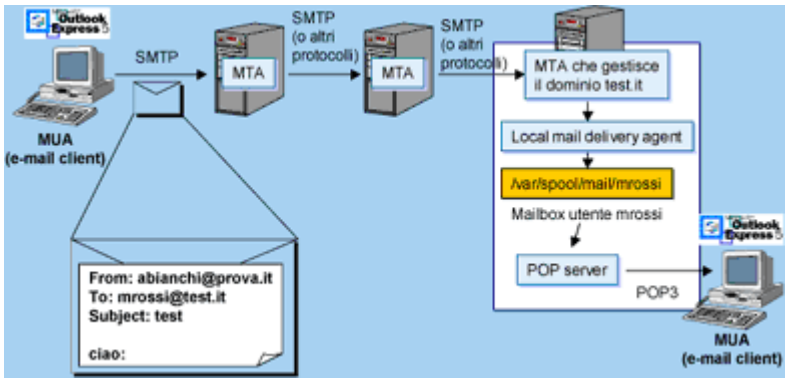
In questo capitolo viene descritto un **sistema di posta elettronica**, soffermandosi sulla funzione dei componenti principali. Viene inoltre descritto il funzionamento del **protocollo SMTP** e vengono introdotti i programmi disponibili in ambiente *UNIX* per la realizzazione di un servizio di posta elettronica.

Struttura e flussi in un sistema di e-mail

Componenti di un sistema di posta elettronica in *UNIX*

Un sistema di posta elettronica per funzionare prevede l'interazione di diversi componenti. I principali sono i seguenti:

- MUA (*Mail User Agent*). Il comune *client* di *e-mail*. È l'interfaccia fra l'utente ed il sistema di posta elettronica. I messaggi in arrivo vengono prelevati dalla *mailbox* dell'utente, residente su un *server*, mentre quelli in uscita vengono passati ad un MTA che si occupa di inoltrarli.
- MTA (*Mail Transport Agent*). Si tratta del componente più importante, il quale svolge diverse funzioni:
  - accetta i messaggi dal MUA o da altri MTA e si occupa di smistarli (*routing*);
  - consegna la posta in uscita verso altri MTA;
  - accetta da altri MTA i messaggi destinati agli indirizzi locali e ne gestisce la consegna nelle *mailbox* degli utenti (*local delivery*);
  - gestisce la posta in transito o in uscita attraverso delle code (*mail queue*).
- MDA (*Mail Delivery Agent*). La spedizione dei messaggi di posta non viene compiuta direttamente dall'MTA ma mediante appositi *Mail Delivery Agent* (o *mailer*), specializzati per i diversi protocolli di trasporto (SMTP, UUCP, consegna locale,...). Anche la consegna dei messaggi per gli utenti locali nelle *mailbox* di sistema avviene utilizzando un apposito *mail delivery agent* (*localmail*, *procmail*).



La figura illustra il tipico percorso di un messaggio di posta elettronica attraverso i componenti descritti.

- Un messaggio di *e-mail*, destinato all'utente mrossi@test.it, viene creato dall'utente abianchi@prova.it mediante un MUA, ad esempio *Outlook Express*, e spedito ad un MTA mediante il protocollo SMTP.
- Lo scambio della posta fra MTA avviene utilizzando un meccanismo di *store and forward* (memorizza e passa avanti), in cui ogni MTA riceve e smista verso un altro MTA i messaggi non destinati ad utenti locali. I messaggi vengono all'evenienza memorizzati per un certo periodo di tempo in una coda, in modo da non perderli qualora uno dei nodi sia temporaneamente inattivo.
- Dopo un certo numero di passaggi, il messaggio giunge all'MTA che gestisce la posta del dominio test.it, dove viene consegnato nella *mailbox* dell'utente locale mrossi e ivi conservato.
- L'utente mrossi preleva i messaggi dalla propria *mailbox* utilizzando un MTU, attraverso il protocollo POP3.

Scelta del MTA

Nel seguito di questa *Unit* faremo riferimento al *software Sendmail*, essendo quello più diffuso e disponibile come standard su tutte le versioni di *UNIX*.

La configurazione di *Sendmail* avviene principalmente mediante il *file* /etc/sendmail.cf, dal formato assai criptico. Nelle versioni recenti non è necessario modificare a mano questo *file*, in quanto esso viene creato a partire da un *file* dal formato più semplice utilizzando un processore di macro. Le difficoltà di configurazione sono tuttavia giustificate dalle ampie possibilità di personalizzazione e dalla ottima flessibilità offerte da *Sendmail*.

Sendmail, specialmente nelle vecchie versioni, è conosciuto come il responsabile di molti problemi di sicurezza. Questa fama dipende in parte anche dal fatto che si tratta di un *software* ad ampia diffusione, e perciò con una ampia base di utenza ed un elevato scambio di informazione e conoscenze. Tuttavia è innegabile che vi siano stati in passato alcuni problemi di sicurezza anche significativi, in parte dovuti alle configurazioni fornite di *default* con alcuni sistemi *UNIX*, in parte derivanti da oggettive motivazioni progettuali, come il fatto che si tratta di un programma eseguibile complesso e funzionante con i permessi di *root*.

Le versioni attuali di *Sendmail*, specialmente quelle fornite con *Linux*, offrono un ragionevole grado di sicurezza e contengono dei meccanismi utili ad evitare abusi (SPAM). Tuttavia, come per ogni altro programma che gestisce un servizio di rete, è buona norma seguire con attenzione i vari bollettini di sicurezza e tenere aggiornato il sistema secondo le indicazioni ivi contenute.

Coloro che desiderassero un MTA meno potente di *Sendmail* ma più semplice da gestire, possono prendere in considerazione una delle seguenti alternative:

- *Smail*. Si tratta di un MTA compatibile con *Sendmail* e con caratteristiche simili. È necessario fare attenzione ad utilizzare una versione recente, in quanto quelle antecedenti alla 3.1.29

soffrono di alcuni problemi di sicurezza. È disponibile con licenza GPL.
- *Exim*. Concettualmente simile a *Sendmail/smail*. Viene fornito assieme ad una ottima documentazione, è efficiente ed offre un notevole controllo contro lo SPAM0.
- *Qmail*. Si tratta di un *software* intrinsecamente più sicuro rispetto agli MTA tradizionali, essendo formato non da un unico programma eseguibile, bensì da componenti più piccoli e pertanto più facilmente controllabili.
- *Postfix*. È un *software* piccolo, veloce, efficiente ed estremamente sicuro, che tuttavia non offre alcune delle funzionalità che invece sono disponibili utilizzando altri prodotti.

MUA

Mail User Agents

Per *UNIX* sono disponibili diversi *client* di posta elettronica, sia funzionanti in modalità testuale (*elm*, *pine*) che con interfaccia grafica. Esistono inoltre dei *software* che consentono la lettura e spedizione dei messaggi attraverso un *Web server* (*openwebmail*).

Il tipico *client* di posta elettronica di solito preleva i messaggi ricevuti da una *mailbox* residente su un *server* mediante il protocollo POP3 (o IMAP). Il MUA di solito non è in grado di occuparsi dell'instradamento dei messaggi, pertanto tutta la posta in uscita viene passata, utilizzando il protocollo SMTP, ad un MTA in grado di compiere l'operazione (*smarthost*).

Nel caso il *client* giri sulla stessa macchina *UNIX* dove risiede la casella dell'utente, questa può essere letta anche accedendo direttamente al *file* corrispondente nella *directory* di *spool* della posta. Nei *client* più semplici, che non supportano SMTP, la posta viene spedita all'MTA utilizzando un programma accessorio con funzione di *mail delivery agent* per il protocollo SMTP.

MIME

La gestione degli allegati nei messaggi di posta elettronica è competenza del MUA. Per i sistemi *UNIX* è disponibile il pacchetto *Metamail*, che contiene i programmi necessari alla creazione e all'elaborazione dei messaggi in formato *MIME* e che può facilmente essere interfacciato con altri programmi che non dispongano di tali funzionalità. Esso può essere utilizzato anche per realizzare *script* per elaborare o gestire messaggi *e-mail* contenenti allegati.

SMTP

Cenni sul protocollo SMTP

SMTP (*Simple Mail Transfer Protocol*) è un protocollo, basato su TCP, che permette di trasferire i messaggi di posta elettronica. Viene utilizzato sia per il collegamento fra MUA e MTA che per lo scambio della posta fra MTA. Per quest'ultimo scopo vengono utilizzati anche altri protocolli come UUCP.

Per quanto riguarda SMTP, la macchina che deve spedire il messaggio di posta elettronica implementa la parte *client* del protocollo, mentre quella che lo riceve funge da *server*. Su di essa deve essere attivo un demone SMTP associato alla porta TCP 25.

È importante notare che SMTP funziona solamente in un verso: ad esempio un MTA non può instaurare una connessione verso un altro MTA per richiedergli la propria posta, bensì può solamente rimanere in ascolto fino a quando l'altro non apre una connessione per spedirgliela. In realtà nelle nuove versioni di SMTP (ESMTP) esiste un meccanismo (ETRN) che permette di chiedere all'altro MTA di svuotare una determinata coda di *e-mail* (anche se logicamente le cose vanno come richiesto, formalmente è comunque l'altro MTA ad instaurare la connessione).

Protocollo SMTP	Descrizione
/usr/lib/sendmail -v mrossi@test.it <messaggio.txt mrossi@test.it ... *Connecting to mail.test.it* via esmtp ...	*Sendmail* viene utilizzato come MDA da linea di comando (in modalità test -v). Mediante una *query* al DNS viene trovato l'indirizzo del MTA che funge da MX per il dominio test.it e viene instaurata una connessione verso esso.
220 *mail.test.it ESMTP Sendmail* 8.11.0/8.11.0; Fri, 29 Mar 2002 20:01:34 +0100 >>> *EHLO* pc001.prova.it 250 - *mail*.test.it *Hello* [243.211.173.213], *pleased to meet you* 250 - *ENHANCEDSTATUSCODES* 250 - *EDITMIME* 250 - *SIZE* 250 - DSN 250 - ONEX 250 - ETRN 250 - XUSR 250 - *HELP*	La macchina che funge da *client* ed il *server* si scambiano informazioni sulla versione del protocollo utilizzato e sulle opzioni disponibili. A questo livello è possibile implementare nel MTA delle restrizioni di accesso basate sull'indirizzo del *client*.

>>> *MAIL From*:beppe@prova.it> *SIZE*=S
250 2.1.0 <beppe@prova.it>... *sender ok*
 	
>>> *RCPT To*:<mrossi@test.it>
250 2.1.5 <mrossi@test.it>... *Recepient ok*
	Il *client* passa al *server* le informazioni sul mittente e sul destinatario del messaggio. Generalmente MTA viene configurato per eseguire delle verifiche sugli indirizzi passati (ad esempio che si tratti di indirizzi per i quali è concesso il *relay* oppure che i valori passati mediante SMTP coincidano con quelli presenti nell'*header* del messaggio (*From* e *To*))

>>> DATA
354 *Enter mail, end with . on a line by itself*
>>> .
250 2.0.0 g2TJIZa07454 *Message accepted for delivery*
beppe@prova.it... *Sent* (2.0.0 g2TJIZa07454 *Message accepted for delivery*)
>>> *QUIT*
221 2.0.0 mail.test.it *closing connection*
	Mediante il comando DATA avviene il trasferimento del corpo del messaggio, che viene passato al *server* in formato ASCII, terminando con una linea contenente un singolo punto. A questo punto il *server* restituisce al *client* un numero di protocollo identificativo del messaggio (che viene inserito anche nell'*header*) e il *client* termina la connessione con *QUIT*.

In figura è possibile vedere le diverse fasi del trasferimento di un messaggio di posta elettronica utilizzando il protocollo SMTP. Per la spedizione è stato utilizzato *Sendmail*, il quale può svolgere le funzioni di MDA se viene lanciato specificando come parametro nella linea di comando utilizzando un indirizzo di *e-mail* valido. Nell'esempio è stato passato in *input* a *Sendmail* un *file* (che dovrebbe essere stato in precedenza adeguatamente preformattato in modo da contenere almeno gli *header* *From*:, *To*: e *Subject*:). È anche possibile scrivere il messaggio direttamente nella linea di comando facendolo terminare con un CTRL-D:

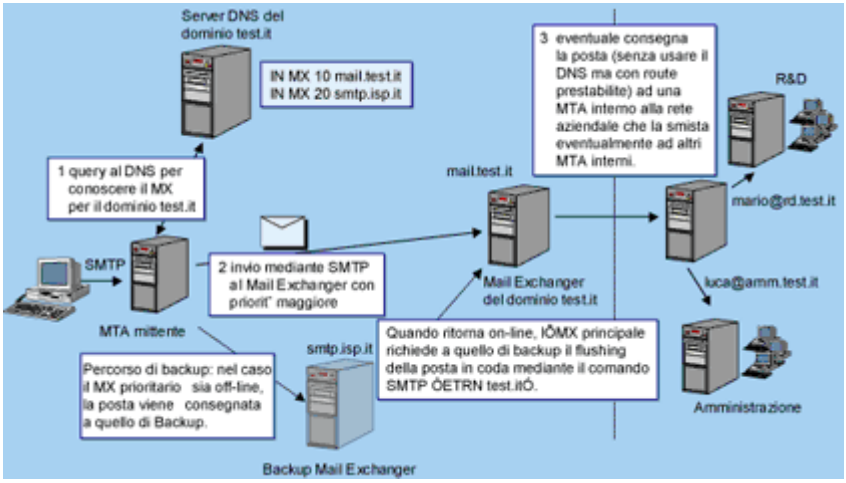
```
# /usr/sbin/sendmail info@test.it
From: Test
To: info@test.it
Subject: prova
```

```
Test
^D
```

Il ruolo del DNS nel protocollo SMTP

Rispetto ad alcuni anni or sono, il *routing* della posta elettronica è molto migliorato in efficienza, specialmente per il fatto che attualmente la maggior parte dei *server* sono perennemente connessi ad Internet. Questo ha permesso da una parte di ridurre i tempi di inoltro dei messaggi, dall'altra di sfruttare la presenza del DNS, che consente di non dover utilizzare (e mantenere aggiornate) complicate tabelle di *routing*.

Il ruolo del DNS non deve essere sottovalutato, in quanto spesso la causa di problemi con la posta elettronica non è dovuta ad una errata configurazione del MTA, bensì a problemi riconducibili al DNS.



Il ruolo del DNS nella consegna della posta è descritto di seguito:

- quando un MTA tenta di spedire un messaggio di *e-mail* ad un determinato destinatario, per prima cosa effettua una *query* al DNS per ricavare l'indirizzo del MTA che funge da *Mail Exchanger* (scambiatore di posta) per l'indirizzo di destinazione. Ad uno stesso dominio possono essere associati più *Mail Exchanger*, per creare una ridondanza oppure per mandare a MTA diversi la posta destinata a sottodomini diversi. Gli indirizzi dei *Mail Exchanger* vengono gestiti mediante dei record di tipo MX nella zona del DNS associata al dominio.
- Una volta ricavati gli indirizzi dei *Mail Exchanger*, l'MTA tenta l'invio del messaggio mediante una connessione SMTP, iniziando dall'MX con priorità maggiore, fino a che la spedizione ha successo. Nel caso la spedizione non vada a buon fine, il messaggio viene tenuto per un certo periodo di tempo in una coda, ritentando l'operazione ad intervalli regolari di tempo.
- Un *Mail Exchanger* può essere il *server* di posta del dominio destinatario (in questo caso il messaggio viene inserito nella *mailbox* dell'utente) oppure può essere configurato in modo da passare la posta ricevuta ad un altro MTA, ad esempio interno alla Intranet aziendale. In questo caso la decisione di *routing* non avviene più mediante una *query* al DNS (che ritornerebbe l'indirizzo del MX stesso), bensì utilizzando un percorso prestabilito.

Quello appena visto è solamente un esempio semplificato che spiega il funzionamento del meccanismo. Nelle pratica si possono trovare diverse soluzioni, dipendenti dalla topologia della rete e dalla politica di gestione del servizio.

Ridondanza e MX di *backup*

Nel caso il *Mail Exchanger* con priorità maggiore sia momentaneamente non disponibile, i messaggi di posta elettronica vengono presi in consegna da un MX secondario, che li tiene in coda e tenta ad intervalli regolari di spedirli al MX principale. Questo avviene utilizzando le informazioni sui record MX prese dal DNS, con alcune cautele, come quelle di non rispedirsi il messaggio e di non creare dei *loop* con altri MX secondari.

In questo modo è possibile realizzare un sistema di *backup*, che, assieme al meccanismo delle code, è in grado di assicurare una certa ridondanza ed affidabilità al sistema.

L'estensione *MIME* ETRN

A questo proposito è importante considerare che se l'MTA di *backup* è configurato in modo da riprovare a spedire la posta in coda a quello primario ad intervalli di tempo troppo lunghi, possono esservi dei rallentamenti anche significativi nella consegna. Nei casi particolarmente sfortunati può anche accadere che per più volte si tenti di rispedire una coda di messaggi proprio nei momenti in cui l'MTA destinatario non è disponibile.

Una possibile soluzione al problema appena incontrato consiste nell'utilizzare il meccanismo di ETRN (*Extended TuRN*), descritto nell'RFC1985.

Il funzionamento è il seguente: quando il *server* principale è disponibile *on-line*, la posta che arriva viene consegnata direttamente ad esso, essendo l'MX con priorità maggiore. Quando invece il *Mail Exchanger* prioritario non è disponibile, la posta viene consegnata a quello di *backup*, che la tiene in coda. Nel momento in cui il *server* principale torna disponibile, esso esegue un collegamento SMTP verso la porta 25 della macchina di *backup* ed impartisce il comando ETRN nomedominio, dopodiché termina la connessione.

Il comando ETRN causa lo svuotamento della coda relativa al dominio indicato da parte del MTA di *backup* verso il MX prioritario. Tentativo che và a buon fine, dato che l'MTA destinatario risulta sicuramente *on-line*.

L'operazione di solito viene eseguita mediante un apposito programma *client*, ad esempio lo *script* `etrn.pl` fornito assieme a *Sendmail*. A scopo didattico può essere interessante simularla a mano mediante telnet:

```
$ telnet smtp.isp.it 25
Trying 145.23.45.67...
Connected to smtp.isp.it.
Escape character is '^]'.
220 smtp.isp.it ESMTP Sendmail 8.9.3/8.9.3; Fri, 5 Apr 2002 13:07:38 +020
0
ETRN test.it
250 Queuing for node test.it started
quit
221 mail.test.it closing connection
Connection closed by foreign host.
```

Sendmail

Sendmail è l'**MTA** più diffuso, in quanto disponibile come standard su tutte le versioni di *UNIX*. Esso offre ampie possibilità di personalizzazione e dalla ottima flessibilità, a scapito di una certa difficoltà di configurazione. Nel corso del capitolo vengono analizzate le varie componenti del sistema e la funzione dei *file* di configurazione, ponendo in particolare l'attenzione sulle problematiche inerenti la sicurezza.

Sendmail può essere utilizzato in differenti modalità, a seconda dei parametri con cui viene chiamato nella linea di comando. La modalità operativa viene selezionata mediante l'opzione -b:

- *sendmail* -bd: funzionamento come demone;
- *sendmail* -bi: *rebuild* del *database* degli alias (*newaliases*);
- *sendmail* -bp: stampa la coda di posta (*mailq*);
- *sendmail* -bt: *test mode* (risoluzione dell'indirizzo senza effettiva spedizione del messaggio).

Specificando invece nella linea di comando un indirizzo di *e-mail*, *Sendmail* può essere utilizzato come un *delivery agent*:

- *sendmail* info@test.it: utilizzo come *mail delivery agent*.

Nel testo verrà utilizzata la scrittura *Sendmail* per indicare il *software* e la scrittura *sendmail* (minuscolo) per indicare il programma eseguibile, che generalmente si trova in `/usr/lib/sendmail` (per brevità il percorso completo non viene indicato).

Analiziamo ora in dettaglio i due utilizzi più frequenti.

Funzionamento come demone (`/usr/lib/sendmail -bd -q5m`)

Utilizzato come *server* SMTP, *Sendmail* è in grado di gestire sulla porta TCP 25 collegamenti SMTP entranti da parte di MUA o altri MTA. Generalmente esso viene avviato come un *server* a sè stante (parametro `-bd`) utilizzando uno *script* di avvio in `/etc/rc.d`. Ovviamente se il sistema non deve ricevere posta dall'esterno mediante SMTP, non è necessario avviare il servizio.

Il parametro `-q5m` indica l'intervallo di tempo che deve intercorrere fra i diversi tentativi di rispedizione di un messaggio che si trova in coda perché i tentativi precedenti di consegna sono falliti.

In *Red Hat Linux* per controllare il servizio si utilizza lo *script* di avvio `/etc/rc.d/rc3/S80sendmail`, richiamandolo con gli opportuni parametri:

- *`/etc/rc.d/rc3/S80sendmail start`*;
- *`/etc/rc.d/rc3/S80sendmail stop`*;
- *`/etc/rc.d/rc3/S80sendmail restart`*.

Funzionamento come *Mail Delivery Agent* (`/usr/lib/sendmail mrossi@test.it`)

Sendmail può essere utilizzato anche come *mail delivery agent* per inviare posta mediante il protocollo SMTP. Di fatto nei sistemi in cui *Sendmail* viene utilizzato come *server* SMTP, esso viene utilizzato anche come MDA per il protocollo SMTP. Quando la copia di *sendmail* in attesa sulla porta 25 riceve un nuovo messaggio non destinato ad un utente locale, essa esegue una nuova copia di *sendmail*, questa volta utilizzato come **MDA**, per il *delivery* via SMTP. Entrambe le copie utilizzano i medesimi *file* di configurazione.

Configurazione di Sendmail

Il *file* di configurazione base di *Sendmail* è:

- `/etc/sendmail.cf`.

Oltre ad esso vengono utilizzati altri *file* per mantenere le informazioni configurabili dall'utente ed i

database (su molte versioni di *UNIX* sono collocati nella *directory* `/etc/mail`). Il nome esatto dei *file* può dipendere dalla particolare installazione e non è detto che nella propria configurazione di *Sendmail* tutte le funzionalità (*features*) che analizzeremo siano attivate.

Nel caso *Sendmail* venga fornito già parzialmente configurato, come avviene per esempio in *Red Hat Linux*, le operazioni di personalizzazione possono nella maggioranza dei casi limitarsi alla gestione dei *file* e dei *database* contenuti nella *directory* `/etc/mail`. Le modifiche più complesse, come l'aggiunta di nuove funzionalità, richiedono invece la modifica del *file* di configurazione globale di *Sendmail*, `/etc/sendmail.cf`, che verrà descritta nei prossimi paragrafi.

I principali *file* di configurazione sono i seguenti:

- `/etc/mail/access` (`/etc/mail/access.db`). *Database* contenente le *access list*. L'argomento verrà discusso nel seguito con maggiore dettaglio.
- `/etc/mail/domaintable` (`/etc/mail/domaintable.db`). *Database* di supporto alla funzione *domaintable*, che facilita il passaggio da un vecchio ad un nuovo nome di dominio, permettendo di utilizzare contemporaneamente più nomi.
- `/etc/mail/local-host-names`. Permette di definire i nomi dei domini che devono essere considerati locali al sistema.
- `/etc/mail/mailertable` (`/etc/mail/mailertable.db`). *Database* di supporto alla funzione *mailertable*, la quale permette di associare ad un nome di dominio un determinato metodo di trasporto. Questo permette ad esempio di reindirizzare tutta la posta di un determinato dominio ad un altro MTA (prova.it smtp:server.pippo.it).
- `/etc/mail/trusted-users`. Lista di utenti che possono spedire messaggi assumendo l'identità altrui senza generare messaggi di errore (*root*, uucp, ...).
- `/etc/mail/virtusertable` (`/etc/mail/virtusertable.db`). *Database* di supporto alla funzione *virtusertable*. Essa permette di mappare domini virtuali in nuovi indirizzi. Si noti che gli indirizzi presenti nell'*header* del messaggio non vengono modificati.
- `/etc/aliases` (`/etc/aliases.db`). *Database* degli alias per gli indirizzi locali (viene ricreato da `/etc/aliases` utilizzando il comando *newaliases*).

Una descrizione più dettagliata delle *features* corrispondenti ai *file* appena descritti verrà effettuata nel seguito.

Database in formato *hash*

I *file* con estensione `.db` contengono *database* in formato *hash*. Essi vengono generati mediante il comando *makemap* a partire da un corrispondente *file* in formato testo. Nella lista precedente sono stati indicati sia i nomi dei *file* di testo sorgente, sia, fra parentesi, i nomi dei *database* risultanti.

Volendo modificare il contenuto del *database mailertable*, si dovranno compiere le seguenti operazioni:

- modifica mediante un *editor* del *file* `/etc/mail/mailertable`.
- Rigenerazione del *database* `/etc/mail/mailertable.db` mediante il comando *`makemap hash /etc/mail/mailertable < /etc/mail/mailertable`*.

Nella *directory* `/etc/mail` è presente un *Makefile* che permette di semplificare tali operazione mediante il comando *make*.

Domini virtuali

Mediante *virtusertable* è possibile definire dei domini virtuali, operazione non possibile

semplicemente inserendo più nome in `/etc/mail/local-domain-names`, in quanto così facendo gli utenti info@test.it e info@prova.com verrebbero mappati sul medesimo utente info.

Mediante *virtusertable* è possibile associare i due indirizzi ad utenti *UNIX* differenti:

info@test.it a.verdi
info@prova.com m.rossi

La ridirezione può avvenire anche verso indirizzi esterni:

info@test.it c.gialli@tin.it
info@prova.com n.verdi@hotmail.com

È inoltre possibile mandare tutta la posta di un dominio nella *mailbox* di un singolo utente:

info@test.it infotest
@test.it d.arancio
sales@prova.com m.verdi
info@prova.com r.rosa
@prova.com s.viola@mail.it
@www.prova.com webmaster
@www.test.it webmaster

Funzionamento di Sendmail

Il funzionamento di *Sendmail* è basato su un *rewriting engine*, il quale, in base all'indirizzo del destinatario, si occupa del *routing* dei messaggi di posta elettronica, restituendo come risultato il *mailer delivery agent* più opportuno per la spedizione del messaggio.

Quando si spedisce un messaggio, l'indirizzo del destinatario viene innanzitutto tokenizzato, ovvero spezzato in parti utilizzando alcuni caratteri speciali come separatori. Ad esempio l'indirizzo mrossi@test.it diventa

- mrossi;
- @;
- test;
- .;
- it;

Successivamente l'indirizzo viene elaborato mediante delle *subroutine* (*rule sets*) secondo il flusso schematizzato in figura. Le *subroutine* da S0 a S5 hanno i seguenti scopi predefiniti:

- S0: restituisce il *mail delivery agent* con cui spedire il messaggio;
- S1: elabora l'indirizzo del mittente;
- S2: elabora l'indirizzo del destinatario;
- S3: preelabora tutti gli indirizzi;
- S4: postelabora gli indirizzi;
- S5: riscrive gli indirizzi degli utenti locali (*unaliased*);
- Snn: *rule sets* specifici per i vari *mailer*;

Ogni *subroutine* è costituita da un insieme di regole di riscrittura (*rules*), descritte nel *file* di configurazione `/etc/sendmail.cf`, come una lista di espressioni regolari con cui di volta in volta viene confrontato l'indirizzo destinatario:

- Rlhs <tab> rhs <tab> commento.

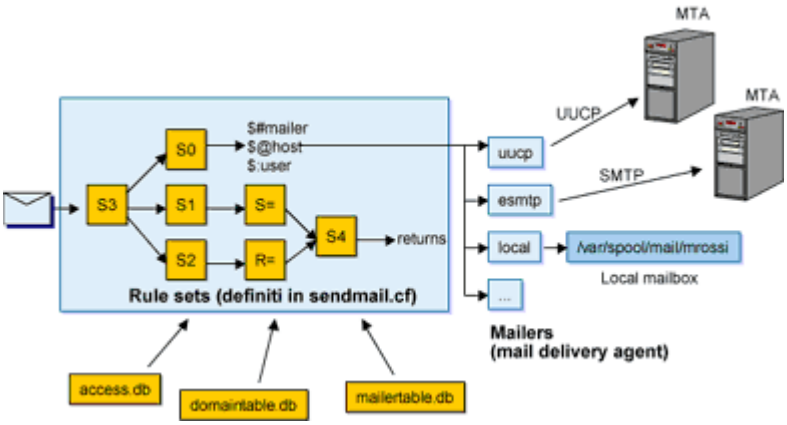
Se l'indirizzo verifica espressione lhs (*Left hand side*), esso viene riscritto secondo quanto specificato in rhs (*Right hand side*).

Oltre alla riscrittura dell'indirizzo, rhs può contenere la chiamata ad una *subroutine* oppure la restituzione di una variabile. In uscita dalla *subroutine* S0 viene restituita una terna che contiene:

- $#*mailer* il nome del *mail delivery agent* da utilizzare per la spedizione (esmtp, uucp, *smarthost*, ...);
- $@*host* l'indirizzo dell'MTA a cui spedire il messaggio;
- $:*user* indirizzo riscritto dell'utente;

Il *file* di configurazione `/etc/sendmail.cf`

La maggior parte delle informazioni necessarie al funzionamento di *Sendmail* sono definite all'interno del *file* di configurazione `/etc/sendmail.cf`. La sintassi utilizzata è abbastanza complessa, perché all'interno di esso sono presenti anche le regole di riscrittura.



In questa sede non entreremo troppo in dettaglio sulla configurazione di `sendmail.cf`, in quanto esiste un metodo più semplice di configurazione, che ne prevede la generazione automatica a partire da un formato più comprensibile. Vale tuttavia la pena conoscerne almeno il formato generale:

Ogni linea di *sendmail* ha uno scopo preciso, identificato dal primo carattere:

- D: definisce una macro;
- C: definisce una *class* (macro contenente una lista di valori);
- F: definisce una *class* leggendo i valori da un *file* oppure dall'*output* di un programma;
- K: esegue una ricerca su un *database*;
- V: indica la versione del *file* di configurazione;
- T: definisce i *trusted users* (utenti fidati);
- P: definisce una priorità di *delivery*;
- O: definisce una opzione;
  - *Character Options*: OL9
  - *String Options*: OLogLevel=9
- H: definisce un *header*;
- M: definisce un *mail delivery agent*;
- S: indica l'inizio di una *subroutine* (*rule set*);
- R: definisce una regola di riscrittura.

Esempi:

Per offrire una idea delle diverse possibilità, di seguito sono descritti alcune linee di configurazione in `/etc/sendmail.cf`. Per maggiori dettagli o per conoscere le altre opzioni si possono studiare direttamente i commenti contenuti all'interno del *file*.

- `Cwlocalhost test.it www.test.it dns.test.it linux.test.it`. Definisce la lista degli *hostname* o nomi di dominio che devono essere considerati locali, ovvero se viene ricevuta una *mail* destinata ad un utente del tipo mrossi@www.test.it, essa viene consegnata all'utente locale mrossi. Nelle versioni di *Sendmail* successive alla V8.7 è possibile utilizzare una scrittura del tipo C{*hosts*} *localhost* test.it www.test.it dns.test.it linux.test.it.
- `Fw/etc/mail/local-host-names`. Ha la stessa funzione della linea precedente, con la differenza che la lista viene letta dal *file* `/etc/mail/local-host-names`.
- `DSmail.tin.it`. Definisce lo *smarthost*. Viene utilizzata per passare in blocco tutta la posta in uscita ad un MTA che si occupi di smistarla (ad esempio al *server* SMTP del *provider* Internet oppure all'MTA centrale dell'azienda).
- `Dmtest.it`. Permette di mascherare il nome dell'*host* nel campo mittente dei messaggi in uscita (utente@pc001.test.it diventa utente@test.it).
- `Kaccess` *`hash`* `-o /etc/mail/access`. Definisce il *database* in formato *hash* da utilizzare per il controllo della *access list*.

Alcune *Options* utili sono le seguenti:

- OLogLevel=9 (oppure OL9): definisce il livello di *logging*;
- Ox12: definisce il carico della CPU oltre il quale i messaggi non devono essere spediti bensì messi in coda;
- OX8: definisce il carico della CPU oltre il quale non accettare ulteriori connessioni;
- OT24h1m: definisce il *timeout*, scaduto il quale un messaggio in coda viene rispedito al mittente;
- OT2d/1h: definisce il *timeout*, scaduto il quale viene notificato al mittente che un suo messaggio è ancora in coda;
- `OQ/usr/spool/mqueue`: definisce la *directory* di *spool* dei messaggi.

Le seguenti macro risultano predefinite in `/etc/sendmail.cf`:

- Dw: nome dell'*host* (pc001);
- Dm: *domain name* (test.it);
- Dj: nome canonico dell'*host* (pc0011.test.it);
- Dv: versione di *sendmail*;
- Dn: nome del mittente da utilizzare nella spedizione dei messaggi di errore (esempio: *Mailer-Daemon*);

Configurazione mediante M4

Un metodo più semplice per configurare *sendmail* consiste nella generazione automatica del *file* `sendmail.cf` a partire da un formato più comprensibile, basato su delle macro che vengono convertite nel linguaggio di `sendmail.cf` utilizzando il processore di macro M4.

Le operazioni da compiere per ottenere un *file* di configurazione di *sendmail* utilizzando m4 sono le seguenti:

- generazione di un *file* (esempio: `file.mc`) in cui siano descritte, sotto forma di macro, le caratteristiche (*features*) desiderate per la propria installazione di *sendmail*;
- trasformazione di `file.mc` nel corrispondente *file* .cf mediante m4 (il percorso del *file* cf.m4 dipende dalla versione di *sendmail* e dal sistema operativo utilizzati):
  `m4 /usr/share/sendmail-cf/m4/cf.m4 <file.mc> > file.cf`

- eventuale test del nuovo *file* di configurazione risultante (`sendmail -Cfile.cf -v -bt info@test.it`);
- installazione del *file* .cf come `/etc/sendmail.cf`;
- riavvio del servizio *sendmail*.

Poiché il *file* sorgente in formato .mc è necessario nel caso si volessero apportare delle modifiche successive, è buona norma tenerlo in una *directory* che non venga cancellata durante un eventuale aggiornamento del sistema. Una buona soluzione potrebbe essere quella di copiarlo in `/etc/sendmail.mc`.

Seppur possibile, è buona norma evitare di apportare modifiche direttamente a `/etc/sendmail.cf`, ma è preferibile imparare come compiere l'operazione attraverso il *file* .mc, essendo esso più indipendente dalla versione di *Sendmail* utilizzata. In caso di aggiornamento del sistema si potrà utilizzare tale *file* per ricreare `/etc/sendmai.cf` utilizzando la versione di cf.m4 fornita col nuovo sistema.

Esempio di configurazione

Di seguito è mostrato il file di configurazione di sendmail tratto da *Red Hat Linux* 9 (redhat.mc).

```
divert(-1)dnl
dnl #
dnl # This is the sendmail macro config file for m4. If you make changes to
dnl # /etc/mail/sendmail.mc, you will need to regenerate the
dnl # /etc/mail/sendmail.cf file by confirming that the sendmail-cf package is
dnl # installed and then performing a
dnl #
dnl # make -C /etc/mail
dnl #
include('/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID('setup for Red Hat Linux')dnl
OSTYPE('linux')dnl
dnl #
dnl # Uncomment and edit the following line if your outgoing mail needs to
dnl # be sent out through an external mail server:
dnl #
dnl define('SMART_HOST','smtp.your.provider')
dnl #
define('confDEF_USER_ID',''8:12'')dnl
define('confTRUSTED_USER', 'smmsp')dnl
dnl define('confAUTO_REBUILD')dnl
define('confTO_CONNECT', '1m')dnl
define('confTRY_NULL_MX_LIST',true)dnl
define('confDONT_PROBE_INTERFACES',true)dnl
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')dnl
define('ALIAS_FILE', '/etc/aliases')dnl
dnl define('STATUS_FILE', '/etc/mail/statistics')dnl
define('UUCP_MAILER_MAX', '2000000')dnl
define('confUSERDB_SPEC', '/etc/mail/userdb.db')dnl
define('confPRIVACY_FLAGS', 'authwarnings,novrfy,noexpn,restrictqrun')dnl
define('confAUTH_OPTIONS', 'A')dnl
dnl #
dnl # The following allows relaying if the user authenticates, and disallows
dnl # plaintext authentication (PLAIN/LOGIN) on non-TLS links
dnl #
dnl define('confAUTH_OPTIONS', 'A p')dnl
dnl #
dnl # PLAIN is the preferred plaintext authentication method and used by
dnl # Mozilla Mail and Evolution, though Outlook Express and other MUAs do
dnl # use LOGIN. Other mechanisms should be used if the connection is not
```

```
dnl # guaranteed secure.
dnl #
dnl TRUST_AUTH_MECH('EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define('confAUTH_MECHANISMS', 'EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN
PLAIN')dnl
dnl #
dnl # Rudimentary information on creating certificates for sendmail TLS:
dnl # make -C /usr/share/ssl/certs usage
dnl #
dnl define('confCACERT_PATH','/usr/share/ssl/certs')
dnl define('confCACERT','/usr/share/ssl/certs/ca-bundle.crt')
dnl define('confSERVER_CERT','/usr/share/ssl/certs/sendmail.pem')
dnl define('confSERVER_KEY','/usr/share/ssl/certs/sendmail.pem')
dnl #
dnl # This allows sendmail to use a keyfile that is shared with OpenLDAP's
dnl # slapd, which requires the file to be readble by group ldap
dnl #
dnl define('confDONT_BLAME_SENDMAIL','groupreadablekeyfile')dnl
dnl #
dnl define('confTO_QUEUEWARN', '4h')dnl
dnl define('confTO_QUEUERETURN', '5d')dnl
dnl define('confQUEUE_LA', '12')dnl
dnl define('confREFUSE_LA', '18')dnl
define('confTO_IDENT', '0')dnl
dnl FEATURE(delay_checks)dnl
FEATURE('no_default_msa','dnl')dnl
FEATURE('smrsh','/usr/sbin/smrsh')dnl
FEATURE('mailertable','hash -o /etc/mail/mailertable.db')dnl
FEATURE('virtusertable','hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
dnl #
dnl # The -t option will retry delivery if e.g. the user runs over his quota.
dnl #
FEATURE(local_procmail,'','procmail -t -Y -a $h -d $u')dnl
FEATURE('access_db','hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE('blacklist_recipients')dnl
EXPOSED_USER('root')dnl
dnl #
dnl # The following causes sendmail to only listen on the IPv4 loopback address
dnl # 127.0.0.1 and not on any other network devices. Remove the loopback
dnl # address restriction to accept email from the internet or intranet.
dnl #
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 587 for
dnl # mail from MUAs that authenticate. Roaming users who can't reach their
dnl # preferred sendmail daemon due to port 25 being blocked or redirected find
dnl # this useful.
dnl #
dnl DAEMON_OPTIONS('Port=submission, Name=MSA, M=Ea')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 465, but
dnl # starting immediately in TLS mode upon connecting. Port 25 or 587 followed
dnl # by STARTTLS is preferred, but roaming clients using Outlook Express can't
dnl # do STARTTLS on ports other than 25. Mozilla Mail can ONLY use STARTTLS
dnl # and doesn't support the deprecated smtps; Evolution <1.1.1 uses smtps
dnl # when SSL is enabled-- STARTTLS support is available in version 1.1.1.
dnl #
dnl # For this to work your OpenSSL certificates must be configured.
dnl #
dnl DAEMON_OPTIONS('Port=smtps, Name=TLSMTA, M=s')dnl
```

```
dnl #
dnl # The following causes sendmail to additionally listen on the IPv6 loopback
dnl # device. Remove the loopback address restriction listen to the network.
dnl #
dnl # NOTE: binding both IPv4 and IPv6 daemon to the same port requires
dnl # a kernel patch
dnl #
dnl DAEMON_OPTIONS('port=smtp,Addr=::1, Name=MTA-v6, Family=inet6')dnl
dnl #
dnl # We strongly recommend not accepting unresolvable domains if you want to
dnl # protect yourself from spam. However, the laptop and users on computers
dnl # that do not have 24x7 DNS do need this.
dnl #
FEATURE('accept_unresolvable_domains')dnl
dnl #
dnl FEATURE('relay_based_on_MX')dnl
dnl #
dnl # Also accept email sent to "localhost.localdomain" as local email.
dnl #
LOCAL_DOMAIN('localhost.localdomain')dnl
dnl #
dnl # The following example makes mail from this host and any additional
dnl # specified domains appear to be sent from mydomain.com
dnl #
dnl MASQUERADE_AS('mydomain.com')dnl
dnl #
dnl # masquerade not just the headers, but the envelope as well
dnl #
dnl FEATURE(masquerade_envelope)dnl
dnl #
dnl # masquerade not just @mydomainalias.com, but @*.mydomainalias.com as well
dnl #
dnl FEATURE(masquerade_entire_domain)dnl
dnl #
dnl MASQUERADE_DOMAIN(localhost)dnl
dnl MASQUERADE_DOMAIN(localhost.localdomain)dnl
dnl MASQUERADE_DOMAIN(mydomainalias.com)dnl
dnl MASQUERADE_DOMAIN(mydomain.lan)dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

I principali tipi di macro presenti nel *file* sono i seguenti:

- le righe inizianti con dnl introducono dei commenti;
- *define*('variabile', 'valore'): definisce il valore di una variabile;
- *FEATURE*('nome','argomenti'): permette di includere in `sendmail.cf` una determinata funzionalità;
- *MAILER*(nome): permette di includere in `sendmail.cf` il supporto per un determinato *mail delivery agent*.

Ogni linea presente nel *file* .mc, quando viene elaborata mediante M4, genera delle linee corrispondenti nel *file* .cf risultante. La trasformazione viene effettuata secondo le regole contenute nel *file* cf.m4 e usando come prototipo il *file* `proto.cf` presente nella stessa *directory*.

Ad esempio la definizione della variabile *SMART_HOST*

```
define('SMART_HOST', 'mail.tin.it')dnl
```

genera nel *file* .cf la linea

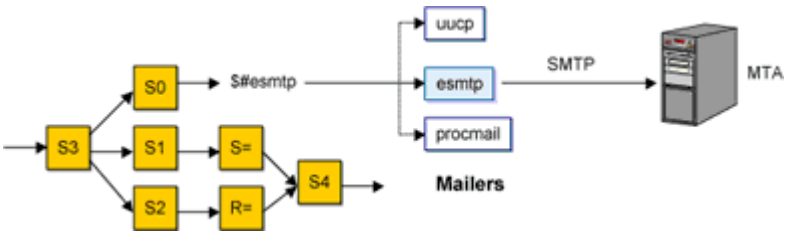
```
Dsmail.tin.it
```

Altre funzioni, ad esempio l'inclusione di un *MAILER* o di una *FEATURE*, possono generare risultati molto più complessi, ad esempio delle regole di riscrittura degli indirizzi.

Per una descrizione dettagliata del formato mc e delle funzionalità che possono essere incluse in *Sendmail* si faccia riferimento al documento *New Sendmail Configuration Files* (`README.cf`), incluso nella documentazione di *Sendmail*.

MAILERS (Mail Delivery Agent)

Per *mail delivery agent* (*mailer*) si intende un programma che si occupa del trasporto di un messaggio di posta elettronica mediante determinato protocollo (SMTP, consegna locale, UUCP, ...). Il nome del *mail delivery agent* da utilizzare per spedire un determinato messaggio viene ritornato come parametro $# in uscita dal *rule set* S0.



È possibile configurare *Sendmail* in modo da utilizzare più *mail delivery agent*, mediante la macro *MAILER*(nome, parametri) nel *file* di configurazione in formato mc:

```
MAILER(procmail)dnl
```

Essa viene convertita nel *file* .cf in linee simili alle seguente, che definiscono il programma e i parametri da utilizzare per il *delivery* della posta locale:

```
Mlocal,
P=/usr/bin/procmail,
F=lsDFMAw5:/|@qSPfhn9,
S=EnvFromL/HdrFromL,
R=EnvToL/HdrToL,
T=DNS/RFC822/X-Unix,
A=procmail -Y -a $h -d $u
```

In una configurazione di *sendmail* possono essere definiti diversi *mail delivery agent*, a seconda delle funzioni e del tipo di protocolli di trasporto della posta che si vogliono supportare. Ad esempio in un MTA di transito in cui si voglia solamente scambiare posta mediante il protocollo SMTP, sarà sufficiente definire:

```
MAILER(smtp)dnl
```

Se si desidera anche consegnare i messaggi per gli utenti locali nelle *mailbox* di sistema, occorrerà definire anche un *local mail delivery agent*:

```
MAILER(local)dnl
```

I *mail delivery agent* più comunemente utilizzati sono i seguenti:

- *MAILER*(*local*).

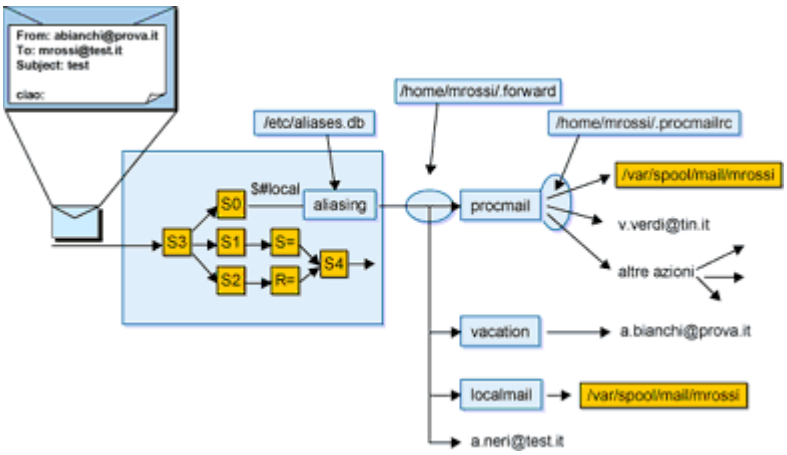
- *MAILER*(prog). I *mailer local* e prog consentono rispettivamente il *delivery* dei messaggi nelle *mailbox* degli utenti locali e l'esecuzione di programmi alla ricezione di un messaggio. Generalmente sono presenti in tutte le installazioni, salvo il caso in cui tutta la posta venga consegnata ad uno *smarthost*.
- *MAILER*(*procmail*). Si tratta di una interfaccia verso il programma *procmail* (che non viene fornito con *sendmail*), utilizzabile come alternativa per il *delivery* locale della posta. *Procmail* viene utilizzato come *default* da molte versioni di *Linux*.
- *MAILER*(smtp). Permette il trasferimento dei messaggi mediante il protocollo SMTP. Se questa macro viene definita, vengono aggiunti quattro *mail delivery agent* con funzioni leggermente diverse: smtp, esmtp (*Extended* SMTP), smtp8 e *relay*. Se si utilizza uno *smarthost* (variabile *SMART_HOST*) viene definito anche il *mailer relay* associato ad esso.
- *MAILER*(uucp). Definisce il *mailer* relativo allo scambio della posta mediante il protocollo UUCP.
- *MAILER*(fax). Permette di spedire messaggi di *e-mail* sotto forma di FAX. Si tratta di una interfaccia verso il programma HylaFAX.

Delivery locale

Un MTA considera come locali, gli utenti con indirizzo del tipo utente@dominio, in cui dominio corrisponda al nome della macchina stessa oppure sia contenuto nella classe Cw. I nomi dei domini locali vengono generalmente letti da un *file* di testo (in *Red Hat Linux* viene utilizzato allo scopo `/etc/mail/local-host-names`):

- miodominio.com;
- miodominio2.com;
- www.miodominio.com;
- mbox.miodominio.com;
- server.miodominio.com;

Quando il destinatario viene riconosciuto come un utente locale, il messaggio viene consegnato all'utente *UNIX* corrispondente alla parte dell'indirizzo di *e-mail* che si trova a sinistra del simbolo @ (è possibile creare degli indirizzi fittizi utilizzando le funzioni di *aliasing*).



La consegna avviene utilizzando il *local mail delivery agent* e, a meno che non siano stati attivati degli alias o *forwarding* su quell'indirizzo, consiste nell'accodare i messaggi nel *file* corrispondente alla *mailbox* dell'utente.

- a.bianchi@miodominio2.com -> a.bianchi -> `/var/spool/mail/a.bianchi`;
- mrossi@miodominio.com -> mrossi -> `/var/spool/mail/mrossi`;
- mrossi@mbox.miodominio.com -> mrossi -> `/var/spool/mail/mrossi`;

Nonostante questa possa sembrare una operazione banale, in realtà vi sono delle complicazioni, come la necessità di gestire il *locking* del *file* per evitare che più programmi (o più copie del *local mailer*) vi possano scrivere contemporaneamente.

Alle volte risulta desiderabile collocare le *mailbox* in un'altra posizione all'interno del *filesystem*, ad esempio all'interno della *home directory* dell'utente *UNIX* corrispondente, allo scopo di poterne fare il *backup* assieme agli altri *file* dell'utente o per gestire un meccanismo di quote su disco. Per far ciò occorre configurare opportunamente il programma associato al *mailer local*. È tuttavia necessario aggiornare anche tutti gli eventuali programmi che gestiscono la posta elettronica, ad esempio il *server* POP3.

Aliasing

Mediante il *file* `/etc/aliases` è possibile definire degli indirizzi locali diversi dai nomi degli utenti *UNIX*.

Il formato di una linea `/etc/aliases` è il seguente:

nome alias: indirizzo/i reale a cui inviare i messaggi ricevuti oppure azione

Mediante il meccanismo degli alias è possibile ad esempio:

- associare più indirizzi al medesimo utente *UNIX* (non viene fatta differenza fra minuscole e maiuscole): Mario_Rossi: mrossi; Mario.Rossi: mrossi; m.rossi: mrossi;
- ridirigere un messaggio ricevuto da un determinato utente ad un altro indirizzo (*forwarding*): *postmaster*: *root*; *sales*: marco@test.it;
- distribuire copia della posta ricevuta su un indirizzo a più utenti: info: beppe, c.verdi@azienda.com, supporto;
- creare delle semplici *mailing list* (per utilizzi più complessi è conveniente utilizzare un *software* opportuno come *Majordomo*): cda: mrossi, averdi, s.bianchi@mail.it;
- scrivere i messaggi di posta elettronica ricevuti in un *file*: *news*: /tmp/news.txt;
- lanciare un programma alla ricezione di un messaggio (passando il messaggio come *input*): mail2www: |/usr/local/bin/mail2www; sms: |/usr/local/bin/smsgw 333123456;
- caricare una lista di alias da un *file*: clienti: :*include*:`/home/averdi/clienti.txt`;

Si noti che è possibile creare un alias che ridiriga verso un altro alias.

Per rendere attive le modifiche apportate al *file* `/etc/aliases` è necessario ricreare il *database* degli alias `/etc/aliases.db` utilizzando il comando *newaliases* (*sendmail* -bi).

Forwarding

Successivamente all'*aliasing*, *Sendmail* verifica se nella *home directory* dell'utente *UNIX* a cui deve essere inoltrato il messaggio di posta elettronica è presente un *file* di nome '`.forward`' (esempio: `/home/mrossi/.forward`). Se questo esiste, vengono eseguite le operazioni in esso descritte, che sono analoghe a quelle effettuabili mediante `/etc/aliases`. Il *file* '`.forward`' può essere gestito autonomamente dai singoli utenti, facendo attenzione ai permessi di scrittura.

Per spedire tutta la posta ricevuta ad un altro indirizzo, è sufficiente scriverlo all'interno del *file* '`.forward`':

```
mario.rossi@prova.it
```

Volendo mantenere una copia dei messaggi anche sulla casella locale si dovrà invece scrivere:

```
\mrossi, mario.rossi@prova.it
```

Il carattere di *backslash* \ davanti al nome dell'utente locale indica a *Sendmail* di non applicare ulteriormente il meccanismo degli alias (per evitare *loop*).

Nel caso si desideri abilitare un risponditore automatico o fare elaborare da un programma il messaggio ricevuto, si dovrà invece indicare all'interno di '`.forward`' il percorso del programma da eseguire, a cui il messaggio verrà passato in *input*:

```
\mrossi, |/usr/ucb/vacation mrossi
```

Protezione dallo SPAM

Protezione da abusi

Un MTA è un programma particolarmente complesso e pertanto soggetto a diversi problemi di sicurezza. Un malintenzionato potrebbe utilizzare un MTA non correttamente configurato per:

- accedere ai dati appartenenti ad altri utenti del sistema;
- produrre malfunzionamenti nel sistema (eccessivo carico, consumo di disco o di banda);
- spedire messaggi con un falso mittente;
- spedire messaggi indesiderati (SPAM) agli utenti del nostro dominio;
- spedire messaggi indesiderati (SPAM) ad altri utenti.

In particolare è bene aver cura di evitare che il proprio *server* venga utilizzato come un *open relay*, ovvero come un MTA mediante cui chiunque in Internet possa spedire messaggi di *e-mail*. Questo col duplice scopo di non lasciare che altri consumino nostre risorse e per evitare di risultare i mittenti di messaggi indesiderati (SPAM), con la possibile conseguenza di avere il nostro MTA inserito in qualche *black list*.

Nelle nuove versioni di *Sendmail* sono presenti diverse funzionalità che permettono di tenere sotto controllo la sicurezza del proprio sistema. Esse sono descritte nel *file New Sendmail Configuration Files* (`README.cf`), incluso nella documentazione di *Sendmail*.

La principali funzioni di *Sendmail* relative alla sicurezza sono le seguenti:

- il *relay* è negato per *default*;
- presenza di un *Access Database* (`/etc/access`);
- possibilità di rifiutare messaggi in base al contenuto degli *header* o del corpo di un messaggio;
- possibilità di fare delle verifiche durante la ricezione di un messaggio mediante SMTP (*check_mail*, *check_rcpt*);
- utilizzo facoltativo del TCP *wrapper* (*libtcpwrap*);
- utilizzo di *procmail* come *local mailer*;
- possibilità di utilizzare una *blacklist* pubblica (RBL) per la verifica del mittente di un messaggio (http://maps.vix.com/rbl/).

Controllo del *relay* della posta

Generalmente un MTA che non sia di transito viene configurato in modo da accettare solamente le seguenti tipologie di messaggi:

- messaggi generati all'interno della propria rete e diretti verso qualunque indirizzo;
- messaggi generati dall'esterno e indirizzati verso utenti locali interni alla propria rete;

Il *relaying*, ovvero la trasmissione di messaggi da una macchina esterna al proprio dominio ad un'altra macchina esterna al proprio dominio, è vietato per *default* nelle versioni di *Sendmail* a partire dalla 8.9. Per tornare al comportamento precedente si deve utilizzare *FEATURE* ('*_promiscuousrelay*').

Le versioni recenti di *Sendmail*, per *default* vietano il *relay* anche nel caso l'indirizzo del mittente ricevuto mediante il comando *MAIL FROM*: durante la ricezione di un messaggio usando SMTP non sia corretto o non sia possibile risalire alla macchina mittente mediante una *query* inversa al DNS.

Access Database

È possibile abilitare il *relay* da parte di determinati utenti, indirizzi IP o domini utilizzando le funzioni di *Access Database* presenti in *Sendmail*. Esse vengono abilitate mediante la funzionalità *FEATURE*('`access_db`'). La macro può accettare un secondo parametro che permette di specificare il tipo e la posizione del *database*: *FEATURE*('`access_db`', '*`hash -o /etc/mail/access`*').

Il *file* `/etc/access` è una tabella contenente una lista di indirizzi con associate le relative azioni da compiere quando si riceve un messaggio con quel mittente. Le possibili azioni sono le seguenti:

- *OK*: accetta il messaggio con il mittente indicato, anche se esso sarebbe stato vietato da altre regole (ad esempio se il nome del dominio non è risolvibile);
- *RELAY*: permette il *relay* di messaggi destinati verso il dominio indicato o ricevuti da esso. Serve come un *OK* implicito per eventuali ulteriori controlli;
- *REJECT*: rifiuta i messaggi il cui mittente o destinatario sia l'indirizzo indicato, emettendo un generico messaggio di errore;
- *DISCARD*: scarta il messaggio utilizzando il *mailer* *`$#discard`*;
- nnn messaggio: restituisce al MTA che sta effettuando la connessione SMTP il codice di errore e il testo specificati. Il codice d'errore nnn deve essere concorde allo standard RFC 821.

Trattandosi di un *database* in formato *hash* è necessario rigenerare il corrispondente *database* quando si apportano delle modifiche al *file* di testo `/etc/access`:

```
makemap hash /etc/mail/access < /etc/mail/access
```

Esempi

```
127.0.0.1 RELAY
192.168 RELAY
192.168.12 REJECT
129.79.6.220 RELAY
test.it RELAY
spammer.com 550 non accettiamo messaggi dal vostro dominio
good.spammer.com OK
john@hotmail.com REJECT
baduser@ REJECT
```

Utilizzando la *feature FEATURE*('*blacklist_recipients*') è possibile verificare anche gli indirizzi del destinatario (utenti o *host* locali):

a.verdi 550 questo utente ha cambiato *mail* in andrea.verdi@test.it
pc001.miodominio.com 550 questo *host* non accetta *mail*
e.neri@server02.miodominio.com 550 utente sospeso

Inoltre tale *feature* rende possibile anche il blocco della spedizione di messaggi da parte di utenti locali che abbiano come destinatari determinati indirizzi:

subscribe@barzellette.com 550 non potete spedire *mail* a questo destinatario

TCP *wrapper*

Le versioni di *Sendmail* distribuite nelle versioni recenti di *Linux* sono configurate di serie in modo da utilizzare il TCP *wrapper*. Questo permette di concedere l'accesso al servizio solamente agli *host* abilitati in `/etc/hosts.allow` e costituisce una ulteriore protezione da usare in associazione a quelle già presenti in *Sendmail*.

sendmail: 127.0.0.1, 192.168., mail.mioisp.it

Per il funzionamento del TCP *wrapper* si faccia riferimento alla *Unit* relativa ai servizi di rete.

Gestione della coda, logging e debugging

Mail queue	**Gestione coda**
/var/spool/mqueue/dfg32HULj12622 */var/spool/mqueue/qfg32HULj12622* */var/spool/mqueue/dfg32HULj346f4* */var/spool/mqueue/qfg32HULj346f4* */var/spool/mqueue/dfg32Hdl439dkk* */var/spool/mqueue/qfg32Hdl439dkk*	*# mailq* *var/spool/mqueue (3 requests)* ----Q-ID---- - --*Size*-- -----Q-*Time*----- ----*Sender/Recipient*--- - g32HULj12622 *(l'asterisco indica un messaggio che sta venendo processato) 12498 *Tue Apr* 2 19:30 iltelefonofisso (*Deferred*: mail@virigilio.it.: *No route to host*) test@virigilio.it g32Hdl439dkk 6 *Fri Apr* 5 10:38 *root* (*host map: lookup (profuso.com): deferred*) beppe@profuso.com g32HULj346f4 55462 *Tue Apr* 2 19:28 beppe (*Deferred*: test.it.: *No route to host*) info@test.it

Gestione della coda di *sendmail*

La coda dei messaggi in transito viene gestita da *sendmail* mediante una coppia di *file* nella *directory* `/var/spool/mqueue`:

- dfXXXXXXXX contiene il corpo del messaggio;
- qfXXXXXXXX contiene le informazioni necessarie alla spedizione, gli *header* e l'esito dell'ultimo tentativo di spedizione.

Quando viene ricevuto un messaggio, *Sendmail* tenta di spedirlo immediatamente a destinazione utilizzando il *mail delivery agent* più adatto. Se la spedizione non và a buon fine, il messaggio viene posto in una coda e la spedizione viene ritentata dopo un certo intervallo di tempo.

L'intervallo fra i tentativi successivi può essere modificato agendo sul parametro `-q[time]` (esempio: `sendmail -bd -q10m`).

Esistono alcuni casi in cui il messaggio viene inserito direttamente nella coda, ad esempio se il *server* è troppo carico (opzione OX), oppure se il *mailer* con cui deve essere spedito è marcato come *expensive* in `sendmail.cf` (utile per evitare che la spedizione immediata di ogni messaggio causi eccessive chiamate nel caso di un MTA su una connessione *dial-up*).

La gestione della coda avviene richiamando *sendmail* con uno dei seguenti parametri nella linea di comando:

- `sendmail -bp`: mostra il contenuto della coda di posta (alias: *mailq*);
- `sendmail -q`: forza la spedizione dei messaggi in coda (*flushing*).

È possibile eseguire un *flushing* selettivo solamente di determinati messaggi della coda, utilizzando i seguenti parametri:

- qRstringa: forza la spedizione dei soli messaggi che contengono stringa nell'indirizzo del destinatario;
- qSstringa: forza la spedizione dei soli messaggi che contengono stringa nell'indirizzo del mittente;
- qIstringa: forza la spedizione dei soli messaggi che contengono stringa nell'identificativo del messaggio.

Ad esempio il comando *sendmail* -qRtest.it forza la spedizione dei soli messaggi il cui indirizzo di destinazione contiene la stringa test.it.

Logging

Il log di tutti i programmi che costituiscono il sottosistema di posta elettronica (MTA, *server* POP3, *server* IMAP, ...) avviene utilizzando la *facility mail* del *logger* sistema (*syslog*). Nei sistemi *Linux* i messaggi relativi alla posta elettronica vengono indirizzati nel *file* `/var/log/maillog`, mediante la seguente linea in `/etc/syslogd.conf`:

```
# Log all the mail messages in one place
mail.* /var/log/maillog
```

Test e *debugging*

Uno degli strumenti di test più utili consiste nell'utilizzare *Sendmail* come *mail delivery agent* sfruttando l'opzione -v. Questo permette una semplice e veloce verifica sulla raggiungibilità di un dato indirizzo:

```
$ /usr/lib/sendmail -v mrossi@test.it
Test
^D
mrossi@test.it... Connecting to mail.test.it. via esmtp...
220 mail.test.it ESMTP Sendmail 8.11.0/8.11.0; Fri, 29 Mar 2002 20:01:34 +0100
>>> EHLO pc001.prova.it
250-mail.test.it Hello [243.211.173.213], pleased to meet you
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<beppe@prova.it> SIZE=5
250 2.1.0 <beppe@prova.it>... Sender ok
>>> RCPT To:<mrossi@test.it>
250 2.1.5 <mrossi@test.it>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
```

```
250 2.0.0 g2TJ1Za07454 Message accepted for delivery
beppe@prova.it... Sent (2.0.0 g2TJ1Za07454 Message accepted for delivery)
>>> QUIT
221 2.0.0 mail.test.it closing connection
```

L'opzione -d nella linea di comando di *sendmail* permette invece di gestire il *debugging* delle regole (*rewriting rules*) presenti nel *file* di configurazione.

- /usr/lib/sendmail -v -bt -dcategoria.livello,categoria.livello;
  - -d0-99.127 massimo livello di *debugging* (-d0-99.99 massimo raccomandato);
  - -d: equivalente a -d0-99.1;
  - -d0.1: mostra solo le informazioni generali;
  - -d21.15: *debug* delle regole di riscrittura degli indirizzi.

Il comando presenta un *prompt* in cui deve essere specificata la regola di partenza e l'indirizzo del destinatario da testare. Vengono restituiti il *mailer* utilizzato per la spedizione ($#), l'indirizzo dell'MTA a cui spedire il messaggio ($@) e l'indirizzo (riscritto) del destinatario ($:).

```
/usr/lib/sendmail -v -bt -d21.15
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 beppe@profuso.com
> rewrite: ruleset 3 input: beppe @ profuso . com
-----trying rule: $@
----- rule fails
-----trying rule: $*
-----rule matches: $: $1 < @ >
...
$3:
rewritten as: $# esmtp $@ profuso . com . $: beppe < @ profuso . com . >
rewrite: ruleset 198 returns: $# esmtp $@ profuso . com . $: beppe < @ profuso .
com . >
rewritten as: $# esmtp $@ profuso . com . $: beppe < @ profuso . com . >
rewrite: ruleset 0 returns: $# esmtp $@ profuso . com . $: beppe < @ profuso .
com . >
```

In un sistema di produzione dove non sia consigliabile eseguire modifiche al *file* `/etc/sendmail.cf`, in quanto in uso, si può fare il *debugging* specificando un diverso *file* di configurazione, mediante l'opzione -Cfile.

Servizio POP3/IMAP

I servizi POP3 e IMAP permettono la ricezione dei messaggi in arrivo da parte di un *client* di posta elettronica. L'installazione del *software* che fornisce detti servizi non presenta particolari difficoltà rispetto all'installazione di un comune *server* lanciato mediante `inetd`, tuttavia è bene prestare comunque particolare attenzione, in quanto nelle versioni standard le *password* viaggiano in chiaro.

Server POP3/IMAP4

POP3 (*Postal Office Protocol*, RFC1939) permette lo scaricamento da parte di un MUA dei messaggi di posta elettronica dalla *mailbox* di un utente *UNIX* su un *server*. In alternativa è possibile utilizzare il protocollo IMAP, che offre maggiori possibilità, anche dal punto di vista della sicurezza, e permette la gestione delle cartelle di posta direttamente sul *server*.

Protocollo POP	**Descrizione**
telnet pop3.test.it 110	

Trying 192.34.33.22... *Connected to localhost.localdomain.* *Escape character is ^].* *+ OK POP3 pop3.mail.it v7.59 server ready*	Per mostrare il funzionamento del protocollo POP3, le operazioni compiute dal *client* possono essere simulate utilizzando telnet per effettuare un collegamento alla porta TCP 110 del *server*. Già a questo livello è possibile introdurre delle limitazioni d'accesso nel *server* utilizzando il meccanismo del *TCP wrapper* (*/etc/hosts.allow*).
user mrossi *+ OK User name accepted, password please* *pass* pippo *+ OK Mailbox open, 58 messages*	Una volta instaurato il collegamento, viene effettuata una autenticazione sull'utente utilizzando il classico meccanismo di *username/password*. Si noti che la *password* viene passata in chiaro.
Retr 1 *+ OK 482 octets* *Date: Fri*, 24 Mar 2002 17:19:18 +0100 CET *From*: Andrea Verdi <a.verdi@test.it> *Subject*: Prova *Content-Length*: 260 Prova di invio messaggio. . *dele 58* *+ OK Message deleted*	Mediante un semplice protocollo, il *client* può scaricare, uno ad uno, i messaggi contenuti nella *mailbox* dell'utente. Dopo avere scaricato un messaggio si può cancellarlo dalla *mailbox* nel *server* utilizzando il comando *DELE*
quit *+ OK Sayonara* *Connection closed by foreign host.*	Il *client* termina la connessione con *QUIT*.

In figura è rappresentato un esempio di utilizzo del protocollo mediante il comando telnet, simulando l'attività di un *client* di posta. POP3 è mappato sulla porta TCP 110.

I *server* per i protocolli POP3 e IMAP forniti di serie con *Linux* (ipop3d, imapd) vengono attivati mediante il supervisore inetd (o xinetd) e non necessitano per funzionare di particolari configurazioni oltre a quelle tipiche dei *server* eseguiti in questa modalità.

Nelle versioni standard del protocollo POP3 la posta e le *password* viaggiano in chiaro, il che può rappresentare un rischio, specialmente se le medesime *password* sono utilizzate anche per altri servizi, ad esempio l'accesso alla *shell* sul *server* come utente *UNIX*. Pertanto è buona norma limitare mediante il TCP *wrapper* l'utilizzo del servizio in modo da evitarne l'accesso a *client* il cui traffico possa passare per reti su cui non si ha il controllo della sicurezza. Nel caso tale limitazione non fosse attuabile, esistono delle estensioni al protocollo che permettono la codifica delle *password* (APOP), che tuttavia non sono supportate da tutti i *client*. Come alternativa è possibile sfruttare un *tunneling* SSL mediante il programma stunnel per realizzare una versione criptata del protocollo.

L'utilizzo di inetd per l'avvio dei *server* POP3 e IMAP può rappresentare un problema di prestazioni nel caso si debba gestire un numero elevato di *mailbox*. Purtroppo il *software* standard fornito con *Linux* non permette il funzionamento in modalità standalone. Esistono altri *server* POP3/IMAP che possono essere utilizzati anche come demoni:

- *Qpopper Qualcomm* 4.0 con supporto TLS/SSL
- *Cyrus IMAP Server*

- *GNUpop*
- *Cubic Circle Cucipop*
- ...

MTA su dial-up

In questo capitolo viene presentato un caso abbastanza frequente, ovvero l'utilizzo di un *server* di posta in una rete non perennemente connessa ad Internet, ad esempio una rete con connessione *dial-up*. Vengono presentate le diverse soluzioni possibili e le problematiche che possono derivare da tale configurazione.

MTA su connessione dial-up

Utilizzo di un MTA non perennemente connesso ad Internet

Nel caso la propria rete non risulti perennemente connessa ad Internet, ad esempio perché si utilizza una connessione *dial-up*, è possibile comunque gestire la posta elettronica del proprio dominio mediante un MTA interno.

La posta in uscita generalmente non rappresenta un problema, in quanto è sufficiente che l'MTA interno abbia la possibilità di eseguire un collegamento SMTP verso l'esterno. La posta eventualmente spedita dagli utenti locali se il *server* non è collegato viene tenuta in coda e può essere spedita utilizzando il comando *sendmail* -q. Per minimizzare la durata delle connessioni *dial-up*, generalmente la posta in uscita non viene smistata direttamente verso i destinatari ma viene spedita in blocco ad uno *smarthost*, di solito l'MTA del *provider*.

Per quanto riguarda la posta in ingresso è possibile appoggiarsi ad un MTA esterno perennemente connesso, ad esempio quello del *provider* o di un fornitore di servizi, configurato come *mail exchanger* in modo da raccogliere la posta per il dominio.

Essa verrà poi trasferita al *server* interno quando esso si collega ad Internet richiedendo il *flushing* della coda di posta al *server* del *provider* mediante ETRN. Nel *server* del *provider* l'instradamento della posta verso il *server* aziendale può essere fornito da una *entry* nella *mailertable*:

- test.it;
- smtp:smtpinterno.test.it.

Questo metodo può essere utilizzato solamente se è possibile rendere visibile all'esterno la porta TCP 25 del *server* interno e se si dispone di un indirizzo IP statico. Si noti che non è necessario rendere visibile il *server* interno a tutto il mondo, bensì solamente al MTA del *provider*. Questo può essere fatto, a seconda della topologia della rete, agendo sul *firewall* aziendale oppure utilizzando le funzioni di *access list* di *Sendmail* (*Access Database* oppure TCP *Wrapper*).

Se il *server* non è visibile dall'esterno oppure non utilizza un indirizzo statico, non è possibile utilizzare SMTP, pertanto è necessario trovare una soluzione alternativa. Alcuni metodi che possono essere utilizzati allo scopo sono i seguenti:

- utilizzo del protocollo UUCP. Il protocollo UUCP permette a *server* interno di instaurare la connessione di scambiare in un'unica operazione sia la posta in ingresso che quella in uscita. Anche se attualmente viene poco utilizzato, si tratta probabilmente della alternativa migliore, in quanto efficiente, affidabile e perfettamente integrata in *Sendmail*. Inoltre permette di non dover esporre il *server* interno;
- *hosting* di una o più caselle *e-mail* POP3 presso il *provider*. Consiste nel creare gli utenti locali del proprio dominio direttamente nel MTA del *provider* e nello scaricare mediante POP3 le

singole caselle di posta elettronica. Si tratta di una soluzione limitata e che non consente una gestione autonoma degli utenti;
- *hosting* di una casella POP3 *catchall* presso il *provider*. Consiste in una casella POP3, sul MTA del *provider*, in cui vengono inseriti tutti i messaggi destinati a qualunque utente del dominio. Sull'MTA del *provider* questo si ottiene inserendo nella *virtusertable* una linea del tipo @dominio.it nomecasella. Una volta prelevata la casella è necessario filtrarla in modo da riconsegnare la posta ai diversi utenti. Tale metodo non è consigliato, in quanto durante le operazioni possono venir perse alcune informazioni contenute nell'*header* dei messaggi;

Fetchmail

Fetchmail (http://www.tuxedo.org/~esr/fetchmail/) permette il prelievo di messaggi di posta elettronica da un *server* utilizzando i protocolli più diffusi POP2, POP3, RPOP, APOP, KPOP, IMAP, e ESMTP ETRN. I messaggi così prelevati possono essere salvati in formato *mailbox* oppure passati ad un MTA per essere inoltrati ad indirizzi locali o remoti. Trattandosi di un programma gestibile mediante linea di comando, *fetchmail* può essere utilizzato per automatizzare il prelievo dei messaggi.

La tipica applicazione di *Fetchmail* consiste nel prelievo di caselle di posta elettronica da un *server* remoto e nella consegna ad utenti locali. Poiché la consegna avviene mediante SMTP all'MTA locale, *fetchmail* si integra perfettamente con il sistema di posta di *UNIX* e permette di sfruttarne tutte le funzionalità (*forwarding*, *aliasing*, *procmail*, ...).

Fetchmail può essere eseguito da un utente creando nella propria *home directory* un *file* di configurazione .fetchmailrc contenente i dati necessari all'accesso alle caselle remote ed il nome dell'utente a cui consegnare la posta. Un esempio di *file* .fetchmailrc è il seguente:

```
poll pop.provider.net proto pop3 port 110
user andrea.verdi with pass pippo is andrea.verdi here
user mario.rossi with pass pluto is mrossi here
```

Il *file* di configurazione può essere creato mediante un *editor* oppure utilizzando il programma fetchmailconf, fornito col pacchetto o mediante una versione recente di linuxconf.

Fetchmail può anche essere utilizzato come un demone che, una volta lanciato mediante uno *script* di avvio in `/etc/rc.d`, rimane attivo nel sistema e si occupa di scaricare le caselle ad intervalli di tempo prestabiliti. È inoltre possibile, dopo aver prelevato una casella di tipo *catchall*, fare in modo che *fetchmail* si occupi di suddividerla opportunamente fra i vari utenti.

Problematiche derivanti dall'utilizzo di una connessione *dial-up*

Le operazioni da compiere per lo scambio della posta, ovvero

- lo scaricamento posta mediante *fetchmail* o UUCP o (*fetchmail* -a);
- lo svuotamento coda di spedizione verso lo *smarthost* (*sendmail* -q);

possono essere eseguite automaticamente ad intervalli prescelti mediante lo schedulatore di sistema (crontab). Questa tecnica prende il nome di *polling*. In un sistema *dial-up* bisogna tuttavia fare attenzione a non utilizzare un intervallo inferiore al *timeout* della connessione ad Internet, per evitare che questa rimanga perennemente attiva. Al contrario, un intervallo di tempo troppo lungo causerà dei problemi di lentezza nella ricezione dei messaggi.

Un altro aspetto a cui è necessario porre attenzione consiste nell'evitare che *Sendmail* faccia traffico indesiderato verso l'esterno, che potrebbe causare connessioni indesiderate da parte del *router*.

Per evitare problemi è necessario adottare due cautele:

- fare in modo che i messaggi destinati all'esterno non vengano spediti immediatamente ma vengano invece tenuti in coda. Per far ciò si devono marcare i *mailer* smtp, esmtp e *relay* come "*expensive*" (costosi);
- evitare che *Sendmail* effettui delle *query* al DNS. Queste avvengono ad esempio in occasione delle verifiche (*check_mail*, *check_rcpt*) effettuate durante la ricezione mediante SMTP dei messaggi generati dai MUA interni.

Le precauzioni da adottare per eliminare queste cause di traffico indesiderato sono spiegate nella FAQ di *Sendmail*.

Nota: un'altra possibile causa di richieste inutili al DNS è l'impossibilità da parte del *server* di risolvere localmente (mediante `/etc/hosts` oppure mediante un DNS sulla rete interna) gli indirizzi dei *client* locali. Questo problema è stato affrontato in forma generale nella *Unit* relativa ai servizi di rete.