

# Postfix—My MTA is better than yours!

Marcin Hłybin

## Wstęp

Zaczynając zabawę z pocztą każdy administrator powinien umieć rozróżnić trzy podstawowe pojęcia: MUA, MTA, MDA. Pierwszy z nich to *Mail User Agent*, który jest programem uruchamianym na desktopie i służy do pisania i odczytywania mejli<sup>1</sup>

jak na przykład *Evolution*, *Microsoft Outlook*, czy *Mozilla Thunderbird*. Tego typu programy łączą się albo do serwera IMAP/POP3 w celu odebrania poczty, albo poprzez protokół SMTP do MTA. Kolejnym magicznym słowem jest tutaj MTA, bowiem oznacza *Mail Transfer Agent*, czyli software do wysyłania i routingu poczty, którego omówieniem się zajmiemy. Ostatnim elementem jest MDA, czyli *Mail Delivery Agent* odpowiedzialny za dostarczanie poczty do skrzynki pocztowej użytkownika, najczęściej w postaci zapisania mejla na dysku w odpowiednim katalogu.

---

<sup>1</sup> Pisownia pochodzi od autora [przyp. red.]

## Widok od strony SMTP i DNS

Naszym ulubionym MTA na chwilę obecną pozostanie Postfix. W dalszej części napiszę jak wygląda, kto go spłodził i dlaczego jest godny uwagi.

Aby poczta mogła przemierzać czeluści Internetu, MTA musi komunikować się z innymi MTA. Standard komunikacji został ustanowiony dawno temu i do dziś odbywa się za pomocą tego samego protokołu SMTP, opatrzonego kilkoma wodotryskami dostosowującymi go do współczesnych czasów i zastosowań. Przykładowa sesja SMTP wygląda następująco (wiersz pochodzące z serwera mają oznaczenie  $\leftarrow$ , wiersze od klienta  $\Rightarrow$ , zaś  $\backslash$  oznacza złamanie wiersza jedynie na potrzeby składu tej publikacji):

```
$ nc mail1.rootnode.net 25
 $\leftarrow$  220 My MTA is better than yours!
 $\Rightarrow$  EHLO bongo.pl
 $\leftarrow$  250-stallman.rootnode.net
 $\leftarrow$  250-PIPELINING
 $\leftarrow$  250-SIZE 30720000
 $\leftarrow$  250-VERFY
 $\leftarrow$  250-ETRN
 $\leftarrow$  250-STARTTLS
 $\leftarrow$  250-ENHANCEDSTATUSCODES
 $\leftarrow$  250-8BITMIME
 $\leftarrow$  250 DSN
 $\Rightarrow$  MAIL FROM: bongo@bongo.pl
 $\leftarrow$  250 2.1.0 Ok
 $\Rightarrow$  RCPT TO: marcin@rootnode.net
 $\leftarrow$  450 4.2.0 <marcin@rootnode.net>: Recipient address rejected:  $\backslash$ 
 $\backslash$  Your message has been greylisted for 300 seconds, please wait.
 $\Rightarrow$  RCPT TO: marcin@rootnode.net
 $\leftarrow$  250 2.1.5 Ok
 $\Rightarrow$  DATA
 $\leftarrow$  354 End data with <CR><LF>.<CR><LF>
 $\Rightarrow$  Hello World..
 $\Rightarrow$  .
 $\leftarrow$  250 2.0.0 Ok: queued as AF2C3655C6
```

W powyższym przykładzie zainicjalizowaliśmy rozszerzoną sesję SMTP. Rozszerzenie to jest nazywane ESMTP od *Extended* (lub *Enhanced*) SMTP. W protokole SMTP pierwszym poleceniem, które wydajemy jest HELO, natomiast jeśli chcemy zainicjalizować sesję ESMTP, korzystamy z polecenia EHLO. W odpowiedzi serwer przedstawia nam rozszerzenia, które obsługuje. Z ważniejszych są to PIPELINING, pozwalający na komunikację bez oczekiwania na odpowiedź po każdym poleceniu; SIZE, definiujący maksymalną wielkość przesyłki oraz STARTTLS obsługujący połączenia szyfrowane.

2xx	wszystko w porządku
3xx	wymagane są dodatkowe parametry
4xx	tymczasowy problem
5xx	krytyczny problem

**Tab 1** Pierwsza cyfra kodu SMTP

My nie spamujemy, dlatego gdy minie 300 sekund, ponownie inicjalizujemy sesję. Tym razem komenda RCPT TO powinna zakończyć się sukcesem i możemy przystąpić do przekazania treści za pomocą polecenia DATA. Zastanówmy się przez chwilę nad specyficznymi kodami: 250 i 450, które po każdym poleceniu otrzymujemy od serwera. Rodzaj kodu zależy od pierwszej cyfry, zgodnie z Tab 1.

Ostatni wiersz naszej sesji informuje nas, że poczta została przyjęta i otrzymała identyfikator AF2C3655C6. Jest on bardzo istotny i pozwala w razie problemów na wyszukanie wiadomości w logach serwera pocztowego.

Pojawia się naturalne pytanie: skąd serwer pocztowy wie, dokąd wysłać pocztę? W tym momencie uśmiecha się do nas usługa DNS i rekordy MX. Wpis MX (Mail eXchanger) określa, który serwer pocztowy jest odpowiedzialny za obsługę poczty dla danej domeny. Jeśli domena takiego wpisu nie posiada, to zgodnie z RFC korespondencja jest pchana do maszyny określonej rekordem A.

Wysyłając pocztę na adres marcin@rootnode.net, klient pocztowy (MUA) komunikuje się za pomocą protokołu SMTP z serwerem pocztowym (MTA). Postfix sprawdza najpierw w konfiguracji, czy sam nie obsługuje domeny rootnode.net. Jeśli tak, to – w naszym przypadku – przekazuje mejla do Dovecota (MDA), który zapisuje go na dysku twardym, w skrzynce użytkownika. Jeśli jednak Postfix nie obsługuje poczty dla tej domeny, odpytuje DNS o rekordy MX dla domeny rootnode.net:

```
;; ANSWER SECTION:
rootnode.net. 300 IN MX 300 mail1.rootnode.net.
rootnode.net. 300 IN MX 300 mail2.rootnode.net.
```

W powyższym przykładzie widzimy, że istnieją dwa serwery obsługujące pocztę dla domeny rootnode.net i mają one identyczny priorytet 300. Jeśli komunikacja z pierwszym z nich została pomyślnie nawiązana, poczta zostanie przekazana właśnie jemu. Jeżeli nie, to Postfix komunikuje się z drugą maszyną. Jeśli jednak tamta także nie odpowiada, to mejl czeka w kolejce do momentu, aż któraś z maszyn zacznie odpowiadać lub skończy się czas przechowywania mejla w kolejce. Domyślnie jest to 5 dni.

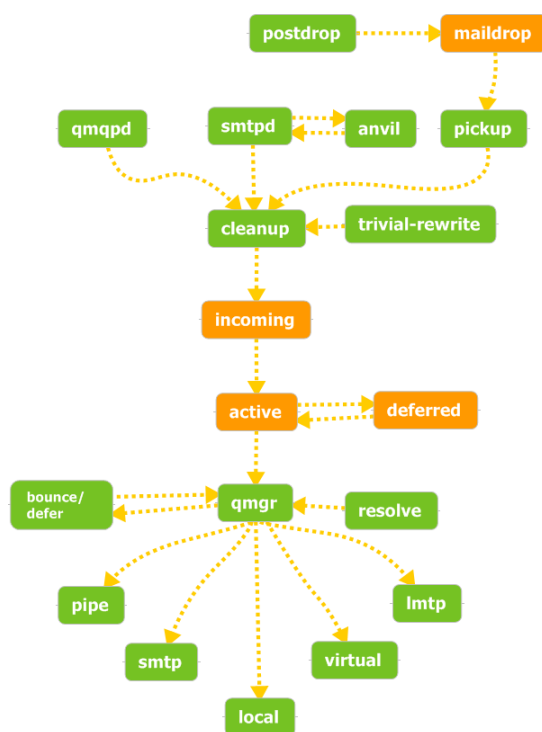
## Postfix

Postfix został napisany przez Wietse Venema. Pierwsza wersja ujrzała światło dzienne w 1999 roku jako półroczny projekt firmy IBM, opatrzony publiczną licencją. Od tej pory, jako doskonała alternatywa dla dużego, monolitycznego serwera poczty Sendmail, pręźnie się rozwija. Postfix został zaprojektowany z myślą o bezpieczeństwie, dlatego też, zgodnie z uniksową filozofią, podzielono go na wiele małych programów,

W dalszej części sesji definiujemy nadawcę oraz odbiorcę poczty. Po komendzie RCPT TO otrzymujemy błąd 450 – efekt działania greylistingu, który odracza przyjęcie poczty na 300 sekund sygnalizując tymczasowy błąd. Jest to zabezpieczenie przed spamerami, którzy hurtowo wysyłają mejle i nie dbają o ponowienie próby.

sterowanych demonem master. Każdy z nich odpowiada za jedno zadanie i jest uruchamiany z minimalnymi uprawnieniami, potrzebnymi do wykonania tego zadania.

Wietse Venema pomimo swojego wieku (ur. w 1951 roku) jest nadal aktywnym developerem. Poruszając na liście dyskusyjnej ciekawy problem, możesz się spodziewać jego odpowiedzi. Może nie każdy wie, ale Wietse jest również twórcą TCP wrappera, dostępnego w każdej dystrybucji GNU/Linux w postaci demona tcpd. TCP wrapper to sieciowy system ACL pozwalający na blokowanie dostępu do usług w warstwie aplikacji, którego konfiguracja odbywa się za pomocą plików `/etc/hosts.allow` i `/etc/hosts.deny`. Dodatkowo, Wietse w 1995 roku wraz z Danem Farmerem, stworzył rewelacyjny skaner sieciowy o nazwie S.A.T.A.N.



**Rys 1** Architektura systemu Postfix

przesyłkę do kolejki maildrop. Następnie demon pickup sprawdza, czy pojawiły się nowe wiadomości. Jeśli tak, pobiera i przekazuje je do demona cleanup. Alternatywnie – gdy dostajemy pocztę z zewnątrz – demon smtpd bezpośrednio przekazuje pocztę demonowi cleanup.

Cleanup robi porządki z otrzymanym mejlem: dodaje brakujące nagłówki takie jak np. Date, From, czy To. Korzysta czasem z innego demona, trivial-rewrite, który dodaje odpowiednie domeny do adresu w przypadku niepełnych adresów e-mail. Po zakończonej pracy, cleanup przekazuje pocztę do kolejki incoming.

## Architektura

Konfiguracja Postfiksa składa się z dwóch plików – `master.cf` oraz `main.cf`. Pierwszy z nich zawiera definicje dla demona master, który jest odpowiedzialny za uruchamianie na żądanie wyspecjalizowanych demonów. W drugim pliku definiujemy wszystkie opcje Postfiksa, takie jak pliki map, obsługiwane domeny, restrykcje itp. Architektura Postfiksa została przedstawiona na Rys 1.

Wydawać by się mogło, że wysłanie mejla to trywialna sprawa. Jak widać na powyższym grafie, nie do końca. Generalnie możemy wyróżnić dwa elementy – procesy i kolejki. Są zaznaczone różnymi kolorami. Kolejka Postfiksa to nic innego jak katalog w `/var/spool/postfix`.

W przypadku wysyłania mejli lokalnie za pomocą programu sendmail<sup>2</sup>, praca Postfiksa rozpoczyna się od postdropa, który wrzuca

<sup>2</sup>Chodzi o wewnętrzny program Postfiksa, a nie o serwer poczty Sendmail

Postfix został zaprojektowany tak, aby przy dużym obciążeniu systemu lub kończącej się pamięci nie dobić kłękającego już serwera. Dlatego właśnie za pomocą specjalnej kolejki `active`, serwer pocztowy dawkuje sobie obsługę napływających mejli, pobierając ich odpowiednią ilość z kolejki `incoming`.

Mejl czeka w kolejce `active`, aż zainteresuje się nim menedżer kolejek `qmgr`. Jest to serce Postfixa, które odpowiada za obsługę i trzymanie w kupie wszystkich kolejek. Pobierając mejla z kolejki `active`, `qmgr` decyduje, jak Postfix powinien go wysłać: czy przekazać do LDA (*Local Delivery Agent*) czy może przesłać dalej za pomocą protokołów SMTP lub LMTP, czy też przekazać jako wejście do innego programu za pomocą demona `pipe`.

Podsumowując, `qmgr` zarządza następującymi kolejkami:

- ◇ `incoming` – tam wrzucane są mejle, które przychodzą do systemu pocztowego;
- ◇ `active` – kolejka, w której znajduje się limitowana ilość aktualnie przetwarzanych mejli;
- ◇ `deffered` – tam trafiają mejle, które nie mogły zostać dostarczone; dostarczenie jest ponawiane po podwojeniu czasu, który upłynął od poprzedniej próby, aż do końca czasu życia kolejki;
- ◇ `corrupt` – tam lądują uszkodzone lub niedające się odczytać pliki;
- ◇ `hold` – do tej kolejki dostają się wstrzymane mejle i leżą w niej tak długo, aż admin ich nie przywróci.

Na tym etapie wiemy mniej więcej jak działa usługa poczty internetowej i jak jest realizowana przez serwer Postfix. Na stronie [postfix.org](http://postfix.org) dostępna jest znakomita dokumentacja zorganizowana w postaci wielu plików HTML. Każda zmienna konfiguracyjna Postfixa to w praktyce odnośnik do strony, która szczegółowo ją opisuje. Dodatkowo, każdy demon i kolejka Postfixa posiada osobną stronę manuala. W razie problemów warto zajrzeć najpierw tam.

## Konfiguracja

Główny plik konfiguracyjny Postfixa to `main.cf`. Wszystkie dostępne zmienne konfiguracyjne można znaleźć po wydaniu polecenia `man 5 postfixconf`. Za pomocą programu `postconf` można wyświetlić domyślne oraz aktualne ustawienia wszystkich zmiennych. Możemy także edytować konfigurację (parametr `-e`) bez konieczności otwierania pliku `main.cf` w edytorze tekstu.

Podstawową konfigurację Postfixa najczęściej wykona za nas system pakietowy danej dystrybucji. Nie zawsze mu to wychodzi, ale przynajmniej mamy dobry szablon pliku konfiguracyjnego. Gdy już doprowadzimy system pocztowy do działania i `mail.log` wskazuje na to, że poczta wychodzi bezproblemowo na świat, możemy zająć się ciekawszymi rzeczami, czyli konfiguracją filtrowania poczty i restrykcjami.

## Kontrola dostępu

Jak już wspomniałem, w sesji SMTP możemy wyróżnić polecenia HELO (lub EHLO), MAIL FROM, RCPT TO oraz DATA. W każdym etapie sesji SMTP istnieje możliwość kontroli dostępu za pomocą opcji `smtpd_helo_restrictions`, `smtpd_sender_restrictions`, `smtpd_recipient_restrictions`, `smtpd_data_restrictions` oraz `smtpd_end_of_data_restrictions`.

Istnieje jeszcze `smtpd_client_restrictions`, który odwołuje się do wszystkich komend klienckich w sesji oraz `smtpd_etrn_restrictions`, który obsługuje restrykcje dla komendy ETRN (w dzisiejszych czasach już praktycznie nieużywanej – służy ona w połączeniach typu dial-up do pobrania plików kolejki dla danej domeny). Wszystkie zmienne dotyczące restrykcji są opcjonalne, z wyjątkiem `smtpd_recipient_restrictions`, która jest wymagana.

Jak widać, istnieje wiele poziomów kontroli dostępu. By dowiedzieć się, jakie parametry może przyjmować każda z nich, należy odwołać się do man 5 `postconf` lub stron [www](http://www.postfix.org/postconf.5.html#smtpd_helo_restrictions), wpisując na końcu URL-a jedną z powyższych restrykcji (np. [http://www.postfix.org/postconf.5.html#smtpd\\_helo\\_restrictions](http://www.postfix.org/postconf.5.html#smtpd_helo_restrictions)). Przykładowa konfiguracja może wyglądać następująco:

```
smtpd_recipient_restrictions =
    check_recipient_access = hash:/etc/postfix/access
        permit_sasl_authenticated
        reject_invalid_hostname
        reject_non_fqdn_hostname
        reject_non_fqdn_sender
        reject_non_fqdn_recipient
        reject_unknown_sender_domain
        reject_unknown_recipient_domain
        reject_unauth_pipelining
        reject_unauth_destination
        reject_multi_recipient_bounce
        permit_mynetworks
        reject_rbl_client sbl.spamhaus.org
        check_policy_service inet:127.0.0.1:10022
        permit
```

Już po samych nazwach można się domyślić, za co jest odpowiedzialna każda z nich. Oczywiście wyraz `reject` w każdej opcji można zamienić na `permit`, jeśli tylko mamy ochotę inaczej sterować blokadami. Dodatkowo możemy definiować własne mapy dostępowe (jak wyżej, w opcji `check_recipient_access`). Jeśli mamy zbiorcze aliasy firmowe, warto wrzucić je na przykład do takiej mapy:

```
$ cat /etc/postfix/access
all@rootnode.net permit_mynetworks, reject
```

Dzięki temu mamy gwarancję, że nikt spoza naszej sieci nie wyśle mejla na adres `all@rootnode.net`. Taki adres w wielu firmach stanowi doskonałe miejsce docelowe

dla spamu. Należy pamiętać, aby po utworzeniu takiej mapy wykonać polecenie `postmap /etc/postfix/access`, w celu utworzenia bazy czytelnej dla Postfixa.

Skoro już o spamerach mowa, to Postfix umożliwia korzystanie z list RBL (Real-time Blackhole List), obsługę SPF oraz podpinanie zewnętrznych serwerów polityki dostępu. Przytoczony wyżej przykład zawiera definicję `reject_rbl_client sbl.spamhaus.org`, co wymusza sprawdzenie hosta, który się do nas łączy, w serwisie Spamhaus. Od strony technicznej wygląda to tak, że wykonywane jest zapytanie DNS, które – jeśli niczego nie zwraca – oznacza, że host nie jest spamerem.

Tutaj mała uwaga - opcja `reject_rbl_client` właściwie powinna być umieszczona w restrykcji `smtpd_client_restrictions`. W praktyce nie ma to jednak żadnego znaczenia, bo restrykcje dotyczące początku sesji SMTP, aż do momentu `RCPT TO` są wykonywane w tym samym momencie, czyli dopiero po `RCPT TO`. Niektóre serwery pocztowe nie są bowiem przygotowane na odrzucenie żądania już na etapie `HELO` lub `MAIL FROM`. Co więcej, wpis w maillogu nie mógłby jednoznacznie wskazać, który z mejli został odrzucony (tzn. od kogo do kogo), gdyby to nastąpiło tuż po komendzie `HELO`.

Na przytoczonym wyżej przykładzie widać, że dalsze kontrole mogą się odwołać do zewnętrznej usługi za pomocą `check_policy_service`. Mogą to być połączenia zarówno po sockecie uniksowym jak i TCP. Po drugiej stronie nasłuchuje demon zazwyczaj napisany w Perlu lub w Pythonie. Rozwiązania takie funkcjonują szybko i sprawnie, i tylko w bardzo dużych systemach może zająć potrzeba przepisania tych demonów na język niższego poziomu, np. C.

W powyższym przykładzie mamy również zaimplementowany greylisting. Usługa polega na wysłaniu klientowi tuż po komendzie `RCPT TO` kodu 450, tak, aby spróbował ponownie później. Dzięki temu jesteśmy w stanie pozbyć się 80% spamerów, którzy wysyłają mejle przez serwery proxy, zombie na losowo wygenerowane adresy mejlowe i nie kwapią się, by powtórzyć transmisję w przypadku otrzymania kodu 450. Oczywiście należy pamiętać, że spamerzy są coraz sprytniejsi i uodpornienie się na greylisting to tylko kwestia czasu. Często spam jest wysyłany przez prawdziwe serwery pocztowe, na przykład dzięki dziurze w aplikacji webowej lub, co zdarza się niezwykle często, przez niepoprawnie skonfigurowany serwer pocztowy, który staje się otwartym przekaźnikiem (ang. *open-relay*), co oznacza, że każdy, bez konieczności uwierzytelnienia, może wysłać z naszego serwera pocztę do dowolnego odbiorcy.

Można stosować dodatkowe zabezpieczenia jak np. Sender Policy Framework (SPF), opierający się na usłudze DNS. Za pomocą rekordów TXT pozwala odrzucać mejle pochodzące z hostów nieupoważnionych do używania danej domeny. Serwer pocztowy sprawdza wówczas pole nagłówka `Return-path` (które czasami nazywa się także `Envelope-sender`) i porównuje z definicją SPF w rekordzie TXT domeny. Jest to bardzo ciekawy i przydatny mechanizm, ale czasami uprzykrza życie, szczególnie z aliasami na obce domeny oraz z plikami `.forward`.

Kończąc definiowanie restrykcji musimy dodać opcję `permit`, która oznacza, że jeśli poczta przeszła bezproblemowo przez wszystkie wcześniejsze warunki, to powinna zostać zaakceptowana. Administrując produkcyjnym serwerem trzeba uważać z dodawaniem nowych opcji, ponieważ może się okazać, że niewłaściwa konfiguracja spowoduje odrzucenie poprawnych mejli. Aby tego uniknąć, powstała opcja `soft_bounce`, która zamienia wszystkie kody 5xx na 4xx, a te jak już wiemy oznaczają tymczasowy problem i ponawianie prób przez zdalne hosty. Opcjonalnie przed daną

opcją w restrykcjach można wstawić `warn_if_reject`, co spowoduje wygenerowanie w logu ostrzeżenia, kiedy warunek restrykcji zostanie dopasowany.

## Filtrowanie

Postfix dostarcza nam silnych mechanizmów do odrzucania szemranych mejli, ale możemy pójść jeszcze o krok dalej. Kolejnym sposobem podnoszącym bezpieczeństwo poczty jest mechanizm jej filtrowania.

Zaczynając od najprostszych, wbudowanych w Postfiksa, mamy do dyspozycji: `header_checks` oraz `body_checks`. Obydwa posiadają rozbudowaną stronę manuala. Są zaimplementowane w demonie `cleanup` i przetwarzają pocztę zanim trafi do kolejki. Tabela checków zawiera serię wyrażeń regularnych oraz przyporządkowaną im akcję. Po pierwszym dopasowaniu wzorca wywoływana jest akcja i Postfix kończy przetwarzanie, przechodząc do kolejnego mejla. Do rodzajów akcji należą `PERMIT`, `REJECT`, `DISCARD`, `WARN`, `HOLD`, `REDIRECT` i kilka innych. Różnica między `header_checks`, a `body_checks`, jak się można domyślić, polega na tym, że pierwszy przetwarza nagłówki, a drugi treść wiadomości. Przykładowa definicja `header_checks` może wyglądać następująco:

w pliku `/etc/postfix/main.cf`:

```
header_checks = regexp:/etc/postfix/header_checks
```

w pliku `/etc/postfix/header_checks`:

```
/^content-(type|disposition):.*name[[:space:]]*=\.*\.(exe|vbs)/
REJECT Bad attachment file name extension: $2
```

Powyższy przykład odrzuci mejle z załącznikami o rozszerzeniu `exe` lub `vbs`, dodatkowo informując o tym nadawcę. Gdy zamiast `REJECT` zastosujemy `DISCARD`, nadawca nie zostanie poinformowany o niedostarczeniu mejla. W przypadku `WARN`, w pliku loga zostanie zapisana informacja o złym załączniku. `REDIRECT` możemy skierować przesyłkę na dowolny adres, a polecenie `HOLD` zatrzymać ją w kolejce `hold`, aż do interwencji administratora.

Możemy również zdefiniować zewnętrzny filtr mejli. Najpopularniejszym jest perlowy `amavisd-new`, który stanowi most pomiędzy Postfiksem a mechanizmem wykrywającym spam (na przykład `SpamAssassin`), skanerem antywirusowym (takim jak `ClamAV`), czy innymi zewnętrznymi programami, jak na przykład `p0f`. Zewnętrzny filtr definiujemy za pomocą opcji `content_filter`. Filtrowanie poczty odbywa się, gdy wiadomość znajdzie się w kolejce. Musimy wyjąć mejla z kolejki, przetworzyć i ponownie wrzucić go do Postfiksa. W pliku `master.cf` trzeba dodać dwa „serwisy”: pierwszy, odpowiedzialny za odpalenie perlowego filtra przez `spawn` (jest to postfixowy odpowiednik `inetd`), oraz drugi, jako demon `smtpd`, odpowiedzialny za ponowne przekazanie mejla Postfiksowi, już po zakończeniu działania filtra.

Poniżej znajduje się wycinek z pliku `master.cf` odpowiedzialny za uruchomienie filtra oraz powrót mejla do kolejki. Konfiguracja odpowiedzialna za uruchomienie filtra:

```
# =====
# service type private unpriv chroot wakeup maxproc command
#           (yes)  (yes)  (yes)  (never) (100)
# =====

scan      unix -      -      n      -      10      smtp
        -o smtp_send_xforward_command=yes
        -o disable_mime_output_conversion=yes
        -o smtp_generic_maps=

localhost:10025 inet n      n      n      -      10      spawn
        user=filter argv=/path/to/filter localhost 10026
```

W miejsce argv należy wstawić pełną ścieżkę do pliku uruchamialnego filtra, który zostanie uruchomiony z prawami użytkownika określonego zmienną user. Konfiguracja odpowiedzialna za ponowne wrzucenie mejla do kolejki:

```
# =====
# service      type private unpriv chroot wakeup maxproc command
#              (yes)  (yes)  (yes)  (never) (100)
# =====

localhost:10026 inet n      -      n      -      10      smtpd
        -o content_filter=
        -o receive_override_options=no_unknown_recipient_checks,no_header_body_checks,no_milters
        -o smtpd_helo_restrictions=
        -o smtpd_client_restrictions=
        -o smtpd_sender_restrictions=
        -o smtpd_recipient_restrictions=permit_mynetworks,reject
        -o mynetworks=127.0.0.0/8
        -o smtpd_authorized_xforward_hosts=127.0.0.0/8
```

Konfiguracja po stronie pliku main.cf ogranicza się do ustawienia parametrów `content_filter` oraz `receive_override_options`. Pilnuje ona, by do filtra trafiły oryginalne, niezmienione przez żaden mechanizm Postfiksa, nagłówki:

```
content_filter = scan:localhost:10025
receive_override_options = no_address_mappings
```

Istnieją jeszcze dwie inne metody filtrowania poczty. Pierwszą z nich, stosowaną przede wszystkim w listach mailingowych, jest obsługa docelowego adresu email przez skrypt perlowy lub dowolny inny plik uruchamialny, poprzez pipe. Można to w prosty sposób zrobić w pliku `/etc/aliases`. Aby plik `/etc/aliases` był w ogóle przetwarzany poczta powinna być obsługiwana przez demona `local`, który jest domyślnym LDA. Prawie na pewno będziemy obsługiwali wiele wirtualnych domen, korzystając przy tym z demona `virtual` albo bodaj najlepszego pozapostfiksowego LDA – `Dovecot`. Jeśli tak się zdarzy, a chcemy, aby konkretny mejl mógł zostać obsłużony przez plik `/etc/aliases`, musimy najpierw przekierować pocztę na skrzynka@localhost lub skrzynka@\$mydestination. W ten sposób dajemy Postfiksowi znać, że skrzynka jest

kontem na lokalnym komputerze i powinna zostać obsłużona przez demona local. Informację, który z LDA jest obecnie używany, znajdziemy w maillogu:

```
Jul 22 00:17:42 stallman postfix/pipe[5161]: D20DD89E8E: to=<marcin@rootnode.net>, relay=dovecot,
delay=10, delays=10/0.01/0/0.02, dsn=2.0.0, status=sent (delivered via dovecot service)
```

Widzimy, że `relay=dovecot`, czyli dostarczenie poczty do skrzynki użytkownika zostało powierzone programowi Dovecot. Plik `/etc/aliases` może wyglądać następująco:

```
/etc/aliases:
users-pl: "|/usr/bin/mlmmj-recv -L /var/spool/mlmmj/users-pl/"
announce-pl: "|/usr/bin/mlmmj-recv -L /var/spool/mlmmj/announce-pl/"
```

Jak widać, skrzynki `users-pl` oraz `announce-pl` to adresy list mailingowych i są *spipowane*<sup>3</sup> z programem `mlmmj-recv`<sup>4</sup> odpowiedzialnym za odebranie poczty i rozesłanie do wszystkich subskrybentów listy. W to miejsce można oczywiście wstawić dowolny skrypt lub program, który będzie robił z pocztą wszystko, na co tylko mamy ochotę.

W Postfixie została także zaimplementowana obsługa protokołu `mlt`, znanego z Sendmaila. Jest to tzw. filtrowanie przedkolejkowe. Internet pełen jest gotowych skryptów i programów korzystających z `mlt`, z których korzysta się także w Postfixie. Więcej na ten temat znaleźć można w dokumentacji.

## Narzędzia

Gdy mamy już uruchomionego i skonfigurowanego Postfixa, musimy jakoś nim administrować i zarządzać. Istnieje szereg wbudowanych poleceń, które pomogą nam manipulować kolejkami. Trzy najważniejsze z nich, dostarczane przez Postfixa to: `postqueue`, `postsuper` i `postcat`.

Za pomocą polecenia `postqueue -p` możemy wyświetlić kolejkę wiadomości, które z jakiegoś powodu jeszcze nie zostały wysłane. Polecenie jest tożsame z `sendmail`owym `mailq`. Dodatkowo możemy opróżnić (ang. *flush*) kolejkę, czyli w trybie natychmiastowym ponowić próbę wysłania wszystkich przesyłek zalegających w kolejce.

Poleceniem `postsuper` usuwamy wiadomości z kolejki, przenosimy do `hold`, itp. Jeśli chcemy wstrzymać przetwarzanie wszystkich wiadomości, wystarczy użyć polecenia `postsuper -h ALL`.

`Postcat` służy natomiast do wyświetlenia konkretnego mejla z kolejki, przy założeniu, że znamy jego ID:

```
postcat -q 03E437F67A.
```

Ciekawym narzędziem jest konsolowe narzędzie `pfqueue`, które napisano z wykorzystaniem biblioteki `ncurses`. Dzięki niemu mamy podgląd całej kolejki mejli, z którymi jest jakiś problem. Możemy szybko przeglądać wiszące w kolejce mejle, zaznaczyć i usunąć.

<sup>3</sup> Określenie autora [przyp. red.]

<sup>4</sup> Chociaż wyraz `recv` wygląda na błąd, jest to poprawna pisownia nazwy programu.

Tak naprawdę brakuje dobrych narzędzi do zarządzania Postfiksem. Każdy administrator korzysta ze swoich skryptów i aliasów. Brakuje na przykład tak fundamentalnej rzeczy, jak usuwanie wszystkich wiadomości z kolejki, zaadresowanych do danego odbiorcy. Powiedzmy, że wysłaliśmy mejle do wszystkich użytkowników a adres jednego z nich jest już nieaktualny lub serwer pocztowy w ogóle zniknął z sieci. W takiej sytuacji, w kolejce może wisieć nawet kilkaset mejli do tego użytkownika. Oczywiście dzięki pipowaniu poleceń uniksowych i językom skryptowym jesteśmy w stanie sobie poradzić z każdym, nawet najbardziej zawiłym problemem. By usunąć z kolejki mejle do konkretnego odbiorcy, dokumentacja komendy `postsuper` zaleca następujące polecenie (teraz uwaga, trzymać się krzesel!):

```
mailq | tail -2 | grep -v '^ *(\' | awk 'BEGIN { RS = "" }
      # $7=sender, $8=recipient1, $9=recipient2
      { if ($8 == "user@example.com" && $9 == "")
        print $1 }
      ' | tr -d '*!' | postsuper -d -
```

Jak widać nie wszystko w Postfiksie jest proste i przyjemne, choć to kwestia napisania odpowiednich narzędzi.

## Duże systemy pocztowe



Nie ma niestety żadnego manuala ani poradnika, jak tworzyć duże systemy pocztowe oparte na Postfiksie. Trzeba poznać mechanizmy rządzące tym serwerem poczty i pomyśleć, w jaki sposób najlepiej go skalować. Ze względu na modułarną budowę, poszczególne części Postfiksa można rozłożyć między maszynami. Najlepiej – zaczynając od odseparowa-

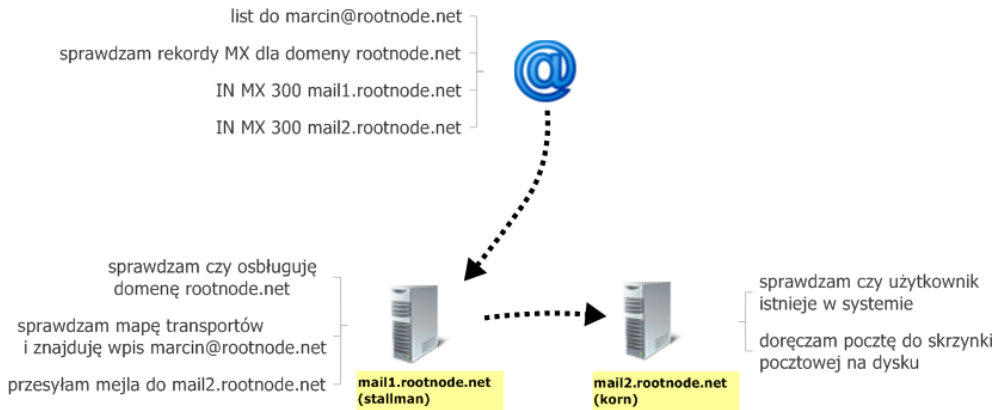
nia serwera poczty wychodzącej od przychodzącej.

Można też poświęcić jedną (lub dwie maszyny w failoverze) na front zajmujący się tylko i wyłącznie routowaniem poczty do wielu maszyn. Jest to popularna metoda przy radzeniu sobie z obciążeniem serwerów www (patrz LVS). Wszystko wygląda pięknie pod warunkiem, że dysponujemy sprzętem odpowiedzialnym za przechowywanie danych (ang. *storage*).

Wówczas wszystkie maszyny stojące za frontem korzystają z jednego zasobu i nie ma konieczności stawiania DRBD z globalnym filesystemem (np. GFS), który bywa niestabilny.

Podobny problem pojawia się w przypadku, gdy użytkownicy korzystający z tej samej domeny pocztowej mają konta na dwóch różnych maszynach. Nie śmiem twierdzić, że Rootnode to duży system pocztowy, ale taki problem się pojawił i trzeba było go rozwiązać. Jak?

Poczta jest przechowywana w katalogach domowych użytkowników. W przypadku jednej maszyny shellowej o nazwie `stallman` problemu nie było – MX dla domeny wskazywał na `stallmana` i poczta działała bezproblemowo. Problem pojawił się po dostawieniu drugiej maszyny shellowej `korn`. Okazało się, że nowi użytkownicy też



**Rys 2** Mechanizm round-robin, przepływ poczty

chcieli korzystać z domeny rootnode.net i też musieli mieć pocztę w katalogach domowych. W efekcie mamy dwa serwery, dwa różne zasoby dyskowe i jedną domenę.

Jeśli pozostawimy MX-y bez zmian, poczta skierowana do użytkownika posiadającego konto na stallmanie zostanie dostarczona, a poczta w tej samej domenie skierowana do serwera korn zostanie odrzucona z komunikatem, że skrzynka nie istnieje.

Otóż wpadliśmy na następujący pomysł: zróbmy na obu serwerach transporty. Na serwerze stallman będzie mapa transportów ze wszystkimi adresami znajdującymi się na kornie. Na kornie zaś będzie mapa transportów ze wszystkimi adresami, które fizycznie znajdują się na stallmanie. Przykładowy wpis w mapie transportów /etc/postfix/transport:

```
marcin@rootnode.net smtp:[89.248.166.201]:25
```

Powyższe oznacza, że wszystko co powinno być dostarczone na adres marcin@rootnode.net wysyłamy protokołem SMTP na port 25 do hosta 89.248.166.201. Kwadratowe nawiasy oznaczają, że jest to docelowy adres IP i nie należy odpytywać DNS o rekord MX.

Należy pamiętać, aby w plikach main.cf obydwu Postfiksów zdefiniować obsługę domeny rootnode.net, czy to w mydestination, czy w virtual\_mailbox\_domains. Ostatnim krokiem konfiguracji jest zmiana rekordów MX dla domeny i nadanie wpisom identycznych priorytetów:

```
$ dig +short rootnode.net mx
300 mail1.rootnode.net.
300 mail2.rootnode.net.
```

Od tego momentu, dzięki algorytmowi round-robin, korespondencja będzie trafiać raz na jeden serwer, raz na drugi.

Zastanówmy się jeszcze przez chwilę nad przedstawionym rozwiązaniem. Rozwiązaliśmy problem obsługi jednej domeny przez dwa serwery pocztowe, bez bramy (ang. *gateway*) na froncie. Zrobiliśmy przy okazji backupowy serwer MX. Dzięki temu,

w przypadku awarii jednej z maszyn, druga przyjmie całą pocztę i będzie przechowywać mejle w kolejce, aż do czasu powrotu drugiej maszyny lub końca życia kolejki (domyślnie 5 dni). Bez problemu możemy też dokładać kolejne maszyny (rysunek 2).

## Na zakończenie

Warto jeszcze wspomnieć gdzie i co czytać. Oczywiście najlepszym źródłem informacji są manuale oraz dokumentacja na stronie `postfix.org`. Jeśli napotkasz problem, który nie został opisany w dokumentacji, albo potrzebujesz porady podczas projektowania swojego systemu pocztowego, warto napisać na listę dyskusyjną `postfix-users`.

Znam tylko dwie książki, które są godne uwagi: *O'Reilly: Postfix the definitive guide* oraz *The book of Postfix: state-of-the-art message transport*. Postfix cały czas zmienia się i ewoluuje. Tworzony jest nowy kod, niektóre funkcje stają się przestarzałe, dlatego nie polecam starszych wydań tych książek. Prawie na pewno spotkasz tam konfigurację POP3, IMAP oraz SASL opartą o oprogramowanie Courier i Cyrus. Ja polecam zastosowanie nowszego, bardzo szybkiego serwera Dovecot, który jest w pełni wspierany przez Postfiksa (od wersji 2.3). Konfiguracja POP3, IMAP i SASL w tym przypadku to kwestia odkomentowania kilku linii w pliku konfiguracyjnym.