RMXにおけるルール群管理機構・エイリアス機構等の実装

小船井　寛†　　青山　陽亮†　　北圍　達也†　　遠山　元道††

† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †{obunai,bluemountain,zonop}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　ルールベースメール配送システム RMX において，先行研究段階では 1 つのルール群の設定しか行うことが出来なかった．そこで本論文ではユーザの使用状況に応じて複数のルール群を管理する手法を提案するとともに RMX を既存の MTA と組み合わせることで信頼性に関する問題も解決した．
また，RMX では動的に柔軟な送信範囲の指定が可能な一方，記述が煩雑になり時間がかかる，アドレスの意味が直感的に分かりづらいなどの問題点があった．このような問題を防ぐため，アドレスに対してエイリアスを登録できる機能を実装した．更に関数形式アドレスにおいて差集合を定義できる差集合号演算を新たに導入した．

キーワード　RMX，電子メール

Implementation of the Rule Set Management and Alias Mechanism in RMX

Kan OBUNAI†, Yosuke AOYAMA†, Tatsuya KITAZONO†, and Motomichi TOYAMA††

† ††Department of Information and Computer Science ,
Keio University
Hiyoshi 3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †{obunai,bluemountain,zonop}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

1. はじめに

RMX(Rule-based e-Mail eXchange system) はユーザが設定したルールとメールアドレスを基に，データベースのアクセスを行い，得られたユーザ集合に対してメールを配信するメール転送エージェントである．先行研究段階での RMX では，1 つのルール群を設定することしかできなかった．また，RMX サーバーに指定できる DNS ドメインも 1 つに限られていた．そこで本論文では，1 台の RMX サーバーにおいて複数のルール群を管理し，複数の DNS ドメインにも対応できる手法を提案した．これにより様々な状況に応じたルール群を作成，管理することが可能となった．また，MTA は電子メールをより安全に正確に転送するという使命や，スパム等の迷惑メールを防がなければならないという責任もある．そのためには，転送エラーメールの再送設定や，不正アクセスの防止等を行わなければならない．しかし，このような複雑な処理を RMX 自体に課すことをさけるため，既存の MTA と RMX を組み合わせるハイブリッドアーキテクチャを提案することで信頼性に関する問題を解決した．
また，RMX では動的に柔軟な送信範囲の指定が可能である一方，アドレスの記述が煩雑になり時間がかかったり，アドレスの意味が直感的に分かりづらくなったりする可能性がある．このような問題を解決するため，ユーザーごとにアドレスに対してエイリアスを登録できる機能を実装した．更に従来の RMX では関数形式アドレスにおいて演算子”+”を用いることでルール間の和集合を定義できたが差集合演算は定義できなかった．そこで関数形式アドレスに限り演算子”-”を用いることで差集合を定義できる機構を新たに導入した．
以下，本稿の構成を示す. まず 2 章で RMX の概要と差集合演算の導入についてを述べる．3 章でルール群管理機構について，4 章でハイブリットアーキテクチャについて，5 章でエイリアス機能について述べ，6 章で RMX のメールの処理能力に関する評価を行い，7 章で結論を述べる．

2. RMX

Rule-based e-Mail eXchange(RMX) は遠山研究室が提案している電子メール配信方式である．一般的にメール配信に利用されているメーリングリスト (ML) ではメーリングリストを代表するアドレスにメールが送信されるとメーリングリストに所属するメンバー全員に対してメールが配信される．

$<ML$ アドレス $>:=<ML$ 名 $>$ @ $<$ ドメイン $>$

メーリングリストではメンバーのメールアドレスの更新など管理者の作業負担が必要となる．例えば大学でのメーリングリストの利用を考える．学生がメールアドレスを変更した場合にはクラス，研究室，プロジェクトなど所属する全てのメーリングリストでアドレスの変更を行わなければならない．

それに対して RMX では下記のような記述により複数の送信先を指定する．RMX ではアドレスの記述方法は基本形式と関数形式の 2 つがあり，これはそのうち基本形式の記述方式を示している．以下，基本形式アドレス，関数形式アドレスの順に説明する．

$<RMX$ のメール配信先指定 $>:=<$ パラメータ $>$ . ... .
$<$ パラメータ $>$ @ $<$ 配送ルール名 $>$ . ... . $<$ 配送ルール $>$
. $<$ サブドメイン $>$. $<$ ドメイン $>$

RMX のメールアドレスは以上のようにサブドメイン以前を表す配送範囲記述部分とドメイン部分が”.”の記号で区分されている．配送範囲記述部は一つ以上のパラメータの組み合わせで構成され，@記号によってパラメータと配送ルール名に分けられる．サブドメインは後述する設定ファイルの名前に相当する．RMX はこのような配送範囲記述を受け取る．そして，指定された配送ルールとそのパラメータに基づきデータベースに問い合わせを行い実際の送信先アドレスを得る．最終的に得られたメールアドレスに基づき配送が行われる．図 1

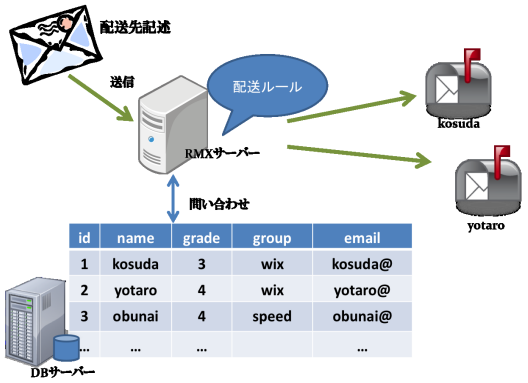

図 1　RMX におけるメール配信の流れ

RMX ではメール配信に必要な情報は全て利用組織のデータベース中で管理されている．そのため，従来のメーリングリストのように一つ一つ登録アドレスを更新する必要はない．また，送信者は配送ルールの記述により柔軟に送信範囲を指定できるため，メーリングリストのようにいくつものグループを用意する必要はない．例えば大学でメーリングリストを作成する場合を想定すると，研究室，学年，学科などの連絡を多く取るであろうグループ毎に，メーリングリストを用意しなければならない．

2.1　配送ルール

配送ルールとは配送範囲記述とそれに基づき送信先メールアドレスを得るクエリを関連付けるルールである．配送ルールは以下のように定義する．

配送ルール名
Type:パラメータの型
query: 送信先メールアドレスを得るためのクエリ

query 部分は SQL によって記述される．RMX は記述された配送ルール名に対応するクエリにパラメータを挿入し，問い合わせを行うことで送信先メールアドレスの集合を得る．このような配送ルールを用いることにより，利用者は簡潔な記述で配送範囲を指定することができる．以下に配送ルール定義の例を示す．

```
grade
gradeType= integer
grade[1]= select s.address from student s
where s.grade = $1 ;
```

上記の例では学年を integer 型で受け取り，それに基づいてメール配信を行う grade ルールを定義している．メールの宛先が 4@grade.example.edu の場合，図 2 のように query 部分で利用者のメールアドレスと学年が格納されている表 student から，学年が 4 年の学生のメールアドレスを得るクエリを記述している．

4@grade.example.edu

id	name	grade	group	email
1	kosuda	3	wix	kosuda@
2	yotaro	4	wix	yotaro@
3	obunai	4	speed	obunai@
...	...	...		...

図 2　grade ルールと配送例

2.2　複数の配送ルールの組み合わせ (基本形式)

ここでは，複数の配送ルールを組み合わせてメールを送る際に，RMX で使用可能な演算子について説明する．演算子を Envelope-To フィールドの配送ルールに使用することにより，ユーザに対してより詳細な配送範囲の指定を可能とする．図 3 では，演算子の詳細を表にしている．

演算子	ターゲット	意味	使用場所	優先順位
.	ルール	積集合	○@○	低
+	パラメータ	和集合	○@×	↕
-		多相	○@×	高

図 3　RMX の演算子

2.2.1 積 集 合

Syntax: $< par_1 >$ . ... . $< par_n >$ @ $< name_1 >$ . ... . $< name_n >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1)\ \cap\ \ldots\ \cap\ name_n\ (par_n)$

“.” は，Envelope-To フィールドに複数の配送ルールを使用したい時に用いられる演算子である．指定された複数の配送ルールの結果の積集合を取り，最終的に得られた結果に対して配送を行う．この演算子を使用するときは，“.” で区分されたパラメータと配送ルール名の数を一致させ，@を挟んでパラメータとそれに対応する配送ルール名を順番に並べる．

例：

toyama.1@lab.grade.lab.example.edu

2.2.2 和 集 合

Syntax: $< par_1 >+\ \ldots\ + < par_n >$ @ $< name_1 >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1)\ \cup\ \ldots\ \cup\ name_1\ (par_n)$

“+” は，ある配送ルールに対して論理和を得たい複数のパラメータを指定するときに用いられる．与えられた複数のパラメータをそれぞれ配送ルールのクエリに代入し，得られた結果の和集合を取る．そして，最終的に得られた結果に対して配送を行う．

例：

3+4@grade.example.edu

2.2.3 多相 (ポリモルフィック)

Syntax: $< par_1 >-\ \ldots\ - < par_n >$ @ $< name_1 >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1,\ ...,\ par_n)$

“-” は，ポリモルフィックな考え方を導入した１つのルールに複数のパラメータを指定するための演算子である．“-” によって与えられたパラメータの数により配送ルールから呼び出すクエリが異なる．次の例ではルール name に２つのパラメータ obunai と kan を与えるため，2 引数の name ルールを使用する．

例：

obunai-kan@name.example.edu

以上の３つの演算子は組み合わせて使用することもでき，ユーザが配送範囲を指定する際の手助けを行う．メールアドレスは以下の形式を取る．

$ListOpe$:= $-$
$UnionOpe$:= $+$
$InterOpe$:= .
Arg := string | integer
$ListPara$:= Arg | $Arg\ ListOpe\ ListPara$
$UnionPara$:= $ListPara$ | $ListPara\ UnionOpe\ UnionPara$
$InterPara$:= $UnionPara$ | $UnionPara\ InterOpe\ InterPara$
$RuleList$:= subdomain | rule $InterOpe\ RuleList$
$Address$:= $InterPara$ @ $RuleList$. domain

例：

obunai-kan+aoyama.toyama@name.lab.example.edu

2.3 関数形式アドレス

関数形式アドレスは以下の形式を取る．

$ListOpe$:= $-$
$UnionOpe$:= $+$
$RuleOpe$:= . | $+$ | $-$
Arg := string | integer
$Para$:= Arg | $Arg\ ListOpe\ Para$
$ParaList$:= $Para$ | $Para\ UnionOpe\ ParaList$
Exp := rule { $ParaList$ }
$ExpList$:= Exp | $Exp\ RuleOpe\ ExpList$
$Address$:= $ExpList$ @ subdomain . domain

学年が 3 年，または 4 年で，名前が obunai である学生へメールを送る場合の例は以下のようになる．

例：

grade{3+4}.name{obunai}@lab.example.edu

関数形式では配送ルール間のユニオンが可能である．例えば学年が 4 年である，または名前が obunai である学生に対してメールを送ろうとした場合以下のように記述すればよい．

例:

grade{4}+name{obunai}@lab.example.edu

また，配送ルール間の積集合をとる’.’ と和集合をとる’+’ が同時に使用された場合には，積集合をとる’.’ の方が優先順位が高いものとする．例えば以下の様なメールアドレスが作成された場合，grade と group の積集合をとった後に，その結果と name との和集合をとる．

例：

name{obunai}+grade{4}.group{speed}@lab.example.edu

2.3.1 ルール間の差分

前述した通り，関数形式アドレスにおいてルール間に演算子”+”を記述することによりルール間のユニオンが可能であった．しかしルール間の差集合は定義する演算子は存在しなかった．そこでルール間に演算子”-”記述することで差集合を定義

できるようにした．例えば学年が 4 年であるが，名前が obunai でない学生に対してメールを送ろうとした場合以下のように記述すればよい．

例:

grade{4}-name{obunai}@lab.example.edu

また，配送ルール間の積集合をとる'.' と差集合をとる'-' が同時に使用された場合には，ルール間のユニオンと同様に積集合をとる'.' の方が優先順位が高いものとする．更に，同一順位の演算子は左から順に評価するため，一度差集合として取り除かれた宛先がその後の記述で再び現れた場合には最終的に送信先の集合にはその宛先も含まれる．例えば以下のようなメールアドレスが作成された場合，speed グループに obunai という学生が含まれていたとしても最終的な送信先の集合には名前が obunai という学生は含まれる．

例:

grade{4}-group{speed}+name{obunai}@lab.example.edu

3. ルール群の管理手法

先行研究段階での RMX では，１つのルール群を設定することしかできなかった．また，RMX サーバーに指定できる DNS ドメインも１つに限られていた．そこで１台の RMX サーバーにおいて複数のルール群を管理し，複数の DNS ドメインにも対応できるルール群の階層化を提案した．

ルール群の階層化にあたり，ユーザーの使用状況を考慮した設計を行った．以下にルール群の管理手法を導入し，各々のルール群の使用状況を挙げる．

ここでルール群とはデータベース情報とそれに対応したルールの組み合わせのことをさす．

- ドメインマスター
  １つの RMX サーバーにおいて１つのルール群のみを使用．
- サブドメイン
  １つの RMX サーバーにおいて複数のルール群を使用．
- マルチドメイン
  １つの RMX サーバーにおいて複数の DNS ドメインを使用．

RMX にはいくつかの設定ファイルが存在する．RMX の動作管理を行う env.properties と，ルール群の管理を行うドメインマスターファイル及びサブドメインファイル，この２種類のファイルをそれぞれの使用状況に応じた記述方式で作成する．env.properties は，ログレベルや、使用ポート，使用できるドメイン，サブドメインの設定を行う．ドメインマスターファイル及びサブドメインファイルでは，ファイル名をそれぞれ“ ドメインエイリアス名.properties ”,“ ドメインエイリアス名_サブドメイン名.properties ”とし，ルール群の設定を行う．1 つのルール群 (SQL クエリ）は共通のデータベースを参照するという制約があるため，使用するデータベースの指定はサブドメインファイルに指定する．前者の RMX 動作管理ファイルは RMX のシステム管理者のみが設定可能であり，後者のドメインマスターファイル及びサブドメインファイルは，ルール群の管理者のみが編集可能である．

3.1 ドメインマスター

ドメインマスターでは，1 つの RMX サーバーにおいて１つのルール群のみを使用する状況を想定している．例えば，企業が１つのデータベースで従業員を管理し，１つの RMX サーバーを運用する場合である．これは先行研究段階の RMX と同じ設計である．

ドメインマスターを使用する際に必要なファイルは，env.properties とドメインマスターファイルのみである．env.properties の domain キーに対し１つのドメインを割り当て，そのドメイン用のドメインマスターファイルを作成することで，RMX を使用することができる．

3.2 サブドメイン

サブドメインでは，１つの RMX サーバーにおいて１つの DNS ドメインに対して複数のルール群を使用する状況を想定している．例えば，企業の部署ごとに従業員を管理しているが，使用する RMX サーバーは企業で１つの場合である．

サブドメインを使用する際に必要となるのは，env.properties と，サブドメインファイルである．env.properties 内の domain キーに使用するドメインを指定し，env.properties 内でそのドメインをキーとして使用するサブドメイン名を指定する．指定されたサブドメインファイルを作成することで，RMX を使用することができる．サブドメインファイルを作成してもドメインマスターファイルを作成することで，ドメインマスターの機能を使用することも可能である．

3.3 マルチドメイン

マルチドメインでは，１つの RMX サーバーに複数の DNS ドメインが割り当てられることを想定している．

マルチドメインを使用する際に必要となるのは，env.properties の domain キーに複数のドメインを割り当てるだけである．使用環境において任意にドメインマスターを使用するか，サブドメインを使用するか，もしくは両方使用するかが選択でき，RMX を使用することができる．

4. ハイブリットアーキテクチャ

RMX と組み合わせる既存の MTA として sendmail を用いた構成を提案する．sendmail は広く普及されている代表的な MTA の１つで，柔軟な設定が可能である．RMX と sendmail を組み合わせた構成が図 4 である．

このように受信側と送信側それぞれに sendmail のデーモンを用いることで，MTA として必要な様々な機能を RMX 自体に持たせることなく実現している．

5. エイリアス機能

エイリアス機能は送信元アドレスごとに，作成した宛先アドレスに対してエイリアスを登録することが出来る．エイリアス機能で用いられるコマンドは store，recall の 2 種類であり，こ

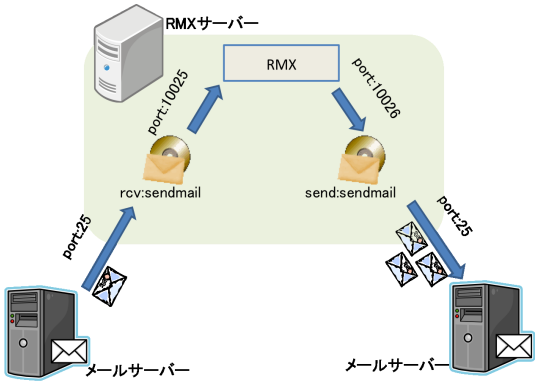

図 4　sendmail を用いたハイブリットアーキテクチャ

れらは宛先にアドレスの先頭にコマンドを加えて利用する．まず，store コマンドを用いてアドレスを登録し，次に recall コマンドで登録したアドレスに対してメールを送信することが出来る．

5.1　store コマンド

store コマンドは以下の形式のように作成したアドレスの前に#で”store. エイリアス名”を挟んで用いる．基本形式，関数形式のアドレス共に使用することが出来る．

*#store.*エイリアス名#登録する送信先アドレス

この形式のアドレスを持つ空メールを RMX に送るとエイリアスの登録が行われる．登録すると送信者に今まで自分が登録したエイリアスとそのアドレスが記載されたメールが送信される．例として store コマンドを用いたときの RMX から返信されたメール本文を図 5 に示す．送信したアドレスは以下の通りである．demo をサブドメインとする．

例：

#store.master#grade{5+6}+name{toyama}@demo.db.ics.keio.ac.jp

```
update succeeded
rmx_alias : master
address : <grade{5+6}+name{toyama}@demo.db.ics.keio.ac.jp>

num |   alias   |              address
-------------------------------------------------------------
1   |master     |<grade{5+6}+name{toyama}@demo.db.ics.keio.ac.jp>
2   |RMX2       |<obunai+bluemountain+zonop@name.demo.db.ics.keio.ac.jp>
3   |RMX        |<grp{speed}-name{yoshda}@demo.db.ics.keio.ac.jp>
4   |test       |<obunai@testo.db.ics.keio.ac.jp>
5   |db22       |<obunai@name.testo.db.ics.keio.ac.jp>
```

図 5　store コマンドによる返信メール本文

RMX は store コマンドを含むメールを受け取ると RMX が作成したデータベースに送信者，エイリアス名，アドレス，サブドメイン，ドメインの組み合わせの行を挿入する．同一の送信者が既にデータベースに存在するエイリアス名等の組み合わせで登録しようとした場合は行の挿入は行われず，アドレスの更新のみが行われる．エイリアスは各ユーザーにつき最大 10 個まで登録可能でそれを超えて登録しようとした場合，一番古いエイリアスが削除される．

5.2　recall コマンド

recall コマンドは以下の形式で用いる．

*#recall.*エイリアス名#@ < *subdomain* > . < *domain* >

recall コマンドで呼び出される送信先アドレスはアドレスに表記されたエイリアス名，サブドメイン名，ドメイン名によって選択される．その後選択されたアドレスに対してメールの送信を行う．

RMX は recall コマンドを含むメールを受け取るとエイリアスの情報を保持するリレーションから送信者，エイリアス名，サブドメイン，ドメインから送信するアドレスを取得しメールを転送する．上記の例で登録したエイリアスを呼び出すには以下のように記述すればよい．このメールは空メールではなく実際に転送されるものなので本文を書くこととなる．

例：

#recall.master#@demo.db.ics.keio.ac.jp

6.　評　　価

6.1　メール処理能力の評価

単位時間あたりに何通のメールを処理できるかについての評価を行う．RMX サーバーのスペックは，CPU に Intel(R)Xeon(R) E5430 2.66GHz，メモリーは 4G，OS は OpenSUSE 11.4，MTA として sendmail 8.14.4 を使用している．

メールアドレスによる転送数を変化させた時，RMX が何通処理できたかを評価する．実験の結果を図 6 に示す．図 6 で見て取れるように，5 件転送アドレス，10 件転送アドレスの場合 1700 通が一分間あたりにおける処理能力の限界だということがわかる．しかし，1 件転送アドレスの場合では，処理能力が毎分 1500 通を超えたところで頭打ちとなっていることがわかる．これは，総転送数が 1500 通の場合，5 件転送アドレスにおいては受信側 sendmail の処理数が 300 通で，送信側 sendmail の処理数が 1500 通であるのに対し，1 件転送アドレスにおいては，受信側 sendmail と送信側 sendmail が共に 1500 通ずつ処理する事となる．これにより，サーバーへの負荷が増加し，sendmail の機能の一つであるロードアベレージが一定値を超えると処理制限を行うという機能が働き，処理数が減少したと考えられる．また，100 件転送アドレスの場合においては，総転送数が 1000 件の時点で毎分あたりの毎分あたりの電子メール処理数が 1000 件未満になっている．これは，RMX の並列処理の手法が原因となっている．RMX の並列処理を実現する際，受信する際に複数のスレッドを立ち上げているが，送信する際には転送数が何件であろうが並列処理でなくシングルスレッドで処理するように設計した．もし，送信する際に転送数に応じて並列処理を行うと，送信側 sendmail へのコネクショ

ン数が激増する恐れがある．コネクション数の激増による送信側 sendmail からの接続制限を回避するため，RMX では送信する際の並列処理を行なっていない．そのため，1 つのアドレスに対し 100 件転送する場合，1 件ずつ転送を行うため全ての転送が終わるまでに約 70 秒必要になり，毎分約 90 通程度の転送しか行えないため，100 件転送アドレスにおける単位時間あたりの処理数が減ってしまったと考えられる．

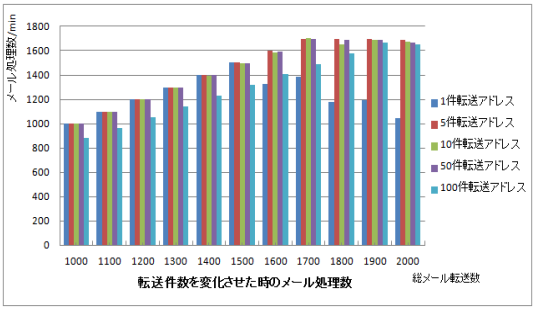

図 6　転送数の変化によるメール処理能力の変化

6.2　エイリアス機能の評価

エイリアスを用いた場合と用いない場合でアドレスの作成時間による評価を行った．使用頻度は低い，普通，高いの３種類，アドレスの長さは 20 文字〜54 文字でアドレス作成にかかった時間の合計を比べる．ここで使用頻度とは使用回数が”低い”は１ヶ月に２回以下，”普通”は１ヶ月に５〜１０回，”高い”は１ヶ月に２０〜３０回とし，それぞれ１ヶ月ごとにランダムで使用回数を決定し６ヶ月分の平均を算出した．

文字数と使用頻度をそれぞれ変化させたときのアドレスの作成時間をまとめた．エイリアスを用いなかった場合の結果を表 1，エイリアスを用いた場合の結果を表 2 に示す．いずれも単位は秒である．

表 1　エイリアスを用いなかった場合のアドレスの作成時間

	20 文字	24 文字	29 文字	34 文字	38 文字	43 文字	54 文字
低い	117	139	151	186	178	218	249
普通	559	650	745	847	889	1316	1566
高い	1982	2218	2432	2718	3004	4234	4939

表 2　エイリアスを用いた場合のアドレスの作成時間

	20 文字	24 文字	29 文字	34 文字	38 文字	43 文字	54 文字
低い	145	127	153	142	152	127	163
普通	604	612	625	606	613	619	621
高い	2018	1959	1982	1996	2017	1999	1952

表 1，表 2 を比べるとエイリアスを用いていない表 1 ではアドレスが長くなるにつれて作成時間は長くなっていることが分かる．一方エイリアスを用いた表 2 ではアドレスの長さに関係なく作成時間が一定であることが分かる．アドレスの長さが短いときにはエイリアスを用いても効果がないことが分かるがアドレスが長いときにはエイリアスの効果が大きいことが分かる．またこれ特徴は使用頻度が高いアドレス程顕著になっていることが分かる．

7.　おわりに

本研究では RMX における複数のルール群を管理する手法，既存の MTA と RMX を組み合わせたハイブリットアーキテクチャの提案を行った．また，アドレスに対してエイリアスを登録する機能および，関数形式アドレスにおいて差集合を定義する演算を導入した．

階層化ルール群による管理手法を導入することにより，様々な状況に応じたルール群を作成，管理することが可能となり，ハイブリッドアーキテクチャとして構成することで信頼性に関する問題を解決した．

また，エイリアス機能では store コマンドと recall コマンドを実装した．これらのコマンドを用いることでユーザーはアドレスに対してエイリアスを登録できるようになり，煩雑なアドレスの入力を防ぐことが出来るようになった．更に差集合演算の導入により今まで不可能だったルール間の差集合の定義が関数形式アドレスに限り可能となり，より柔軟な送信範囲の指定が可能となった．

文　　献

[1] 高畑 理, 藤沼 健太郎, 石橋 玲, 遠山 元道. ”Magic Mirror Mailing：個人情報データベースを利用する柔軟なメイル配送システム”, 情報処理学会データベースシステム研究報告 Pages: 123-128 July 2001

[2] Kim Hanki, Sang-Gyu Shin, Motomichi Toyama. ”A Rule-Based Mailing System for an Organization”, International Workshop on INformation Processing over Evolving Networks, June 2006

[3] 原田 哲志, 慎 祥揆, 遠山 元道. “ RMX における電子メール送受信範囲管理方式の提案”, DBWS2007

[4] 青山 陽亮, 遠山 元道. “ RMX におけるポリモルフィックルールとメール本文編集機能の導入”, DEIM2010

[5] 北圃 達也，青山 陽亮, 遠山 元道. “ RMX における関数形式アドレスおよびデバッグ支援機能の実装”, DEIM2011