セキュリティ対策のための 大学内一括管理向けDNS登録システムの開発

An Implementation of DNS Registration System with Concentrated Management in University for Network Security

山守 一徳, 平田 和之
Kazunori YAMAMORI, Kazuyuki HIRATA

三重大学 教育学部
Faculty of Education, Mie University

〒 514-8507 三重県津市上浜町 1515
1515 Kamihama-chou,Tsu,Mie 514-8507,Japan

Tel:059-232-1211,yamamori@edu.mie-u.ac.jp,e200341@edu-s.cc.mie-u.ac.jp

学内に点在するDNSサーバは，各部局の管理者によって管理されているが，度重なるセキュリティホールの発見によりパッチ当て作業などが頻繁に発生し，管理者の負担となっている．それら複数のDNSサーバを１台のDNSサーバに集約管理することによって，各部局の管理者の負担を減らすだけでなく，セキュリティ強化に繋がると考えられる．その場合，集約管理する管理者の負担も減らすためには，DNS登録作業については各部局の管理者によって行われることが望ましい．筆者らは集約されたDNSサーバの中への登録作業をWEBブラウザを用いて各部局の管理者が分散して行えるシステムを開発した．このシステムは，DNS登録せずにIPアドレスを割り当てたマシンの登録も同時に行うことができる．末端利用者からのIPアドレス申請をWEBブラウザを用いて行なうこともできる．また，学内の全マシンの情報がデータベースの中に登録されるために，情報処理センターのような対外接続管理部局において，セキュリティ対策上必要となる学内全マシンの情報一元管理が可能となるシステムである．ここでは開発したシステムを紹介しその評価について述べる．

キーワード: DNS, 登録システム, ネットワークセキュリティ, Web ブラウザ, Java

The DNS servers which exist scatteringly in a university are managed by administrators of each section. Recently the patch works for the DNS servers are needed frequently because the security holes are found many times. Those works are bothering the administrators. It is considered that the concentrated management from those many DNS servers to one makes the total loads of administrators reduced and makes the network security enhanced. In that case, in order to reduce the administrator's load of the concentrated system, it is desired that administrators of each section perform the DNS registration of the individual section. Thus, We have developed the system that each administrator can perform the DNS registration to the concentrated DNS server using each Web brauzer respectively. And all machines, which are assigned to IP address without DNS registration, can be registered simultaneously. Users can request the assignment of IP address from each Web brauzer. Moreover, because all machine's information are registered in the database, the network administration section like an information processing center can manage the total information of all machines in the university. It leads to enhance the network security. This paper introduces the developed system and shows the evaluation.

Keywords: DNS, Registration System, Network Security, Web brauzer, Java

1 はじめに

大学のネットワークは，各部局に各セグメントを割り振り，セグメント単位に分散管理されている場合が多い．同様にドメイン管理についても，各部局で分散管理され，一つの研究室に一つのドメインを割り当てている場合が多い．そのため，学内にはDNS サーバが，多数存在することとなり，各 DNS サーバは，部局あるいは研究室の単位で，それぞれの管理者が管理している．メールサーバや WWW サーバについても，部局あるいは研究室ごとに管理されている．これらサーバが多数存在し，分散管理されている状態では，最近のネットワーク攻撃に対し，リスクが多数存在する状態にある [1]．このリスクを小さくするためには，サーバの台数自身を少なくすることが効果的である．

これまでは，サーバを集約させることは，その集約されたサーバの管理者に負担が増えるため，分散管理する策を採用することが多かった．サーバの管理は，管理者である教官のボランティア作業によって成り立っており，その教官の負担を少なくするためにも分散管理が採用されたのである．しかし，最近のネットワーク攻撃によって，マシンに被害が出た場合，大学全体のセキュリティ対策が脆弱であると非難されるため，分散管理方式から集中管理方式に移行することが考えられる．

DNS サーバに限らず，各種サーバの設定ファイルを直接編集することなく，GUI を用いた簡単設定で設定することができる商用ソフト [2] が存在するが，そのソフトは，ミドルソフトのバージョンが固定されるという弊害があるだけでなく，主にインストール作業を容易にするためのものである．

また，IP アドレス申請を末端の利用者からメールで受け付けると管理者が DNS 登録作業を行うことを支援するのみのシステム開発例 [3] や Web ブラウザを用いた DNS 管理システムの開発例 [4, 5] があるが，DNS の知識がなくても IP アドレス登録ができることを目的としていたり，ホスト情報の利用者による確認を目的としており，今回のようにセキュリティ対策上から DNS サーバの集約を目的としたものでなく，各部局に管理者が存在し，複数の管理者が自分の管理領域部分にのみ設定ができるという運用形態に向いたものではない．そのため，独自に Java サーブレットを用いてシステムの開発を行った．

本システムは，bind のインストール作業について容易にするシステムではなく，インストールに関しては従来とほぼ同様である．その後の運用形態が，各部局の管理者と分散管理することを目指したシステムであり，集約された 1 台の DNS サーバに対して複数の管理者が各自の管理領域のみ扱うことができるようにしたシステムである．

さらに，本システムは，DNS 登録せずに IP アドレスを割り当てたマシンの管理も同時に行うことができ，これまで DNS 登録とは別に IP アドレスの発行管理がされてきたが，その管理も一緒に行うことができる．末端の利用者が IP アドレス申請を Web ブラウザから行なうと管理者へメール発信されると共にデータベースの中に仮登録され，メールを受け取った管理者が IP アドレス割り当てを行うと共にデータベースに本登録することもできる．また，学内の全マシンの情報がデータベースの中に登録されるために，情報処理センターのような対外接続管理部局において，セキュリティ上必要となる学内全マシンの情報一元管理が可能となる．本論文では，この開発したシステムについて紹介すると共に，実際のデータを投入して行った評価について述べる．

なお，管理者の負担削減のためには Dynamic DNS を導入することも考えられるが，セキュリティ上固定登録して学内のマシンを管理しようとしている方策と相反するため，導入する計画は今のところない．

2 DNS サーバのかかえる問題点

2.1 パッチ当て作業

サーバを運用する場合，これまで稼動開始後に発生する作業としては，通常のデータ更新作業以外に，電源断のトラブル対応やハードウェア故障のトラブル対応が多かったが，近年は，セキュリティホールが発見される度に必要となるパッチ当てまたはバージョンアップ作業に多大の時間を要するようになってきている．DNS の bind 関連ソフトウェアのセキュリティホールが頻繁に発見されているだけでなく，サーバに標準インストールされているソフトウェアのセキュリティホールも頻繁に発見されているため，多種多様なセキュリティホールの情報に対して，対応すべきセキュリティホールと対応しなくてもよいセキュリティホールを切り分け，さまざまな作業を行わなければならない．これらは，サーバ管理を本

職としない教官の管理者にとって苦痛な作業であり，確実にセキュリティホールの対策が行えたのか不安が残る作業である．情報処理センターのような対外接続管理部局にとっても，学内のサーバのセキュリティホール対策が全部実行されたのか確信が持てない状態になっている．この状態を解決するには，サーバを集中管理する形態に変更する方が得策である．

2.2 発生費用

サーバを運用する場合，その初期導入のためのハードの購入費用が発生するだけでなく，運用中においてもハード故障が発生するため，その修理費用も発生する．また，セキュリティ対策のために OS のバージョンアップも必要となる場合があり，そのためのバージョンアップ費用が発生する場合もある．全学で複数のサーバが稼動しているため，これらを積算すると多大な金額になっているはずである．一方，最近のハードウェアは性能が優れ，集約して運用したとしても性能的に問題がないと思われる．よって，発生費用を抑えるためには，サーバの台数を削減すれば良い．

2.3 DNS 登録作業

DNS の設定ファイルの中のレコード表現は，多種多様なことが記述でき，管理者のスキルが要求される．IP アドレスの登録を一つ追加する作業の場合には，定型化された作業で可能であるため，習得すれば可能ではあるが，設定ファイルを直接編集したり，プロセス再起動を行ったりするため，管理者は熟練した特定の人に限られているのが実情である．サーバ管理作業を本職としない教官である管理者の負担を減らすには，IP アドレスの登録作業も年毎に交代することが望まれている．そのためには，IP アドレスの登録作業をもっと簡単化できる必要がある．これは，設定ファイルを直接編集することなく，登録の追加・修正・削除が容易にできるインタフェースを開発すれば解決できる．

また，DNS サーバの台数を単純に集約させるだけの場合，その集約されたサーバの中のドメインまたはセグメントことの設定ファイルを各部局の管理者に直接編集してもらう方法が考えられるが，編集作業の排他制御が困難であり，他の部局の管理範囲まで読み書きができすぎたりするという問題がある．読み書きの制限を部局の管理範囲ごとに限定するには，ファイルのアクセス制限を細かく設定することも考えられが，煩雑であるため好まれていない．これに対しては，ドメインまたはセグメントことの設定ファイルを直接編集させずに，アクセス用システムを開発し，そのシステムで管理者の管理外のドメインやセグメントの中の設定内容までは見えないように制御することで，分散管理をしている場合と同様の情報開示状態にすることができる．

2.4 ２重管理

DNS 登録せずに IP アドレスを発行する場合がある．その管理表は DNS サーバ内の登録されたデータと合わせて２重管理されている場合がほとんどである．部局によっては，２重管理の手間や不一致をなくすために，全てのマシンを DNS 登録してしまうところすら存在するが，セキュリティ対策上，DNS 登録する必要のないマシンまで DNS 登録してしまうことは好ましいことではない．この２重管理の手間をなくすためには，データベースを使って一元管理すると解決できる．

3 データベースと融合する場合の問題点

3.1 データ反映のための再起動

DNS の設定ファイルは，変更するだけでは，効果がすぐに現れない．リフレッシュ時間を待たずしてすぐに効果を出すためには，ネームサーバ（in.named）のプロセスを再起動させる必要がある．そのため，DNS 登録をデータベースと融合させて，データベースの中の登録データを追加・修正・削除した場合，その内容をまず DNS の設定ファイルに反映させ，設定ファイルの中身を追加・修正・削除した後に，in.named のプロセスの再起動までを行わせる必要がある．

また，設定ファイルの中の SOA レコードには，シリアル番号と称する整数値が存在し，設定ファイルに何らかの変更を加えた場合，このシリアル番号の数字を増加させて，変更があったことを示す必要がある．この in.named のプロセスの再起動とシリアル番号の数字をアップさせる作業まで管理者が簡単

にかつ着実に実行できるユーザインタフェースを構築しなければ，真に使いやすいシステムになっているとは言えない．

そこで，Web ブラウザからボタンを押すだけで，シリアル番号の数字が 1 加算された数字に変わり，かつ in.named のプロセスの再起動に相当する操作を行うことができるように設計した．

3.2 多種な DNS レコード記述

DNS の設定ファイルには，SOA レコード，A レコード，PTR レコード，CNAME レコード，MX レコード，NS レコードなど多種類の記述の仕方が存在する [6]．データベースと融合させる場合，IP アドレスとマシン名の対は，A レコードや PTR レコードに相当し，その対をデータベースで管理させることは容易であるが，それ以外については，記述の仕方が多数あり，不向きである．そこで，DNS の設定ファイルの前半部分と後半部分に分け，後半部分は，A レコードまたは PTR レコードとし，前半部分にはそれ以外の SOA レコード，CNAME レコード，MX レコード，NS レコードなどを記述することとした．そして，前半部分は，ドメインまたはセグメントごとに一つのファイルとして自由記述ができることとし，後半部分は，データベースの中に IP アドレスとマシン名の対を登録させ，そのデータから自動生成させることとした．

すなわち，A レコードと PTR レコードはデータベース中の IP アドレスとマシン名の対から自動生成させ，前半部分のファイルと合体して DNS の設定ファイルを生成させることとした．

幸いにも DNS の設定ファイルは省略の記法を使わなければ順番に依存せずにレコード記述することができる．もしも，省略の記法が使われていたとしても容易に補うことができる．そのため，前半部分と後半部分に分割する方法を採用することができた．

3.3 登録作業

データベースを用いる場合，IP アドレスごとに多くの情報を保存することになるため，データ登録作業が大変な作業となる．これに対しては，末端利用者が IP アドレスの申請を出す時に登録データのほとんどをデータベースに入れてもらえるようにした．末端利用者が IP アドレス申請を Web ブラウザから行なうと管理者へメール発信されると共にデータベースの中の仮登録テーブルに挿入され，メールを受け取った管理者は仮登録テーブルの内容を見て IP アドレス割り当てを行うとシステムが本来の IP アドレスごとのデータのテーブルへ移動させることを行なう．なお，末端利用者が Web ブラウザを持ってない場合は紙による申請となるため，管理者が手入力して登録する機能も持つようにした．

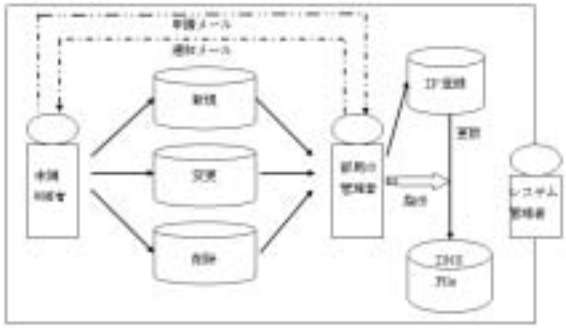

図 1: 作業の流れ

4 システム概要

4.1 利用形態

集約サーバの管理者は，主にソフトウェアのパッチ当て作業を行なう．各部局の管理者は，従来通りのセグメントまたはドメインごとの IP アドレス発行管理および DNS 登録作業を行なう．末端利用者は，IP アドレス新規申請や変更申請や削除申請を行なう．各部局の管理者は末端利用者からの各種申請をメールで受け取ると該当する作業を Web ブラウザを通して実施し，データベース内の更新を行なう．作業の流れを図 1 に示す．

IP アドレス登録データが削除しても良い状態になっても放置されたままになることがよくある．これらのデータの陳腐化を防ぐため，一括して IP アドレス登録データの内容を全利用者に確認しなければならない時がある．この作業のために，データベース内の全データを EXCEL ファイルへ保存する機能を有する．

また，末端利用者が IP アドレスの登録内容を確認したい場合がある．このため，末端利用者は IP アドレス登録データを直接変更することはできないが参照することはできる．末端利用者が変更したい場合には，変更申請を Web ブラウザを通して行な

う．削除申請も同様である．

このシステムにログインして利用するユーザアカウントの権限としては，このシステムの管理者のレベルと各部局の管理者のレベルと末端利用者のレベルの３種類が存在し，さらに，ログインして利用するユーザアカウントごとに，参照することのできるセグメントの範囲を限定する．一方，組み合わせて利用可能なセグメントとドメイン名の対のデータを持つテーブルが用意されており，セグメントを指定すれば利用することのできるドメインが限定することができるため，セグメントの範囲を限定すると同時にドメインの範囲も限定する．

4.2 実現方法

各部局の管理者や末端利用者からは，Web ブラウザを用いて操作できることを目指し，Java サーブレットを用いて開発を行った．また，データベースにアクセスする部分は JavaBeans を用い，表示画面部分は JSP を用いた．JavaBeans を用いることで，データベースをアクセスする SQL 文付近のプログラム部分を容易にはユーザが見ることができないようになっている．メールを発信する部分は JavaMail を用いた．CGI と比べると Java サーブレットは一度呼び出されるとスレッドを常駐するためオーバヘッドが少ないという長所がある．Java サーブレットでなく PHP を用いて開発することも考えられたが，開発したシステムを将来に亘ってメンテナンスしていくためには，Java を利用した方が，メンテナンスできる人材が将来多く存在するであろうと見越して判断した．

Java サーブレットコンテナとしては，Tomcat-4.0.3 を採用し，Apache-1.3.27 および Sun-Microsystems 社の Java 2 Platform Standard Edition(J2SE)-1.4.0 を用いた．さらに，データベースとしては，PostgreSQL-7.2.1 を採用し，JDBC2 を用いた．DNS ソフトとしては，bind-8.3.3 を用いた．用いた OS は，Solaris8 であり，ハードウェアは，SUN Blade 100 である．

4.3 データベーステーブル

データベース内に構築した主なテーブルは以下のものがある．

(1) IP アドレスごとのマシン情報を格納するテーブル

主キーは IP アドレスを元にした 9 桁の整数値である．DNS 設定ファイルの A レコードと PTR レコードは，この中の IP アドレスとマシン名のデータから生成される．

(2) 設定ファイルの存在場所を格納するテーブル

主キーはセグメント名またはドメイン名の文字列である．DNS の設定ファイル並びに前半部分のファイルの存在場所が，絶対パスで示されている．

(3) セグメントとドメイン名の対を格納するテーブル

主キーはセグメントとドメイン名の対である．

(4) ユーザごとのパスワードなどを格納するテーブル

主キーはログインアカウントの文字列である．

(5) 末端利用者から管理者へ新規申請データを渡すためのテーブル

主キーは時刻を元にした識別番号の整数値である．

(6) 末端利用者から管理者へ変更申請データを渡すためのテーブル

主キーは時刻を元にした識別番号の整数値である．

(7) 末端利用者から管理者へ削除申請データを渡すためのテーブル

主キーは時刻を元にした識別番号の整数値である．

(8) 各部局ごとの申請先メールアドレスを格納するテーブル

主キーは部局名の文字列である．

4.4 IP アドレス管理

このシステムは，DNS 登録せずに IP アドレスを発行したマシンの管理にも用いることを目指し，データベースのレコードの中の項目に，正引きと逆引きのそれぞれの DNS 登録を行うか否かのフラグ

図 2: 新規入力画面の例

図 3: 申請確認画面の例

を持つこととした．DNS 登録を行わない場合には，データベースの中に登録されるだけになり，DNS の設定ファイルへ反映されることはない．

また，セキュリティ対策上，情報処理センターのような対外接続管理部局において，学内の全マシンの情報が参照できることが望ましいため，データベースのレコードの項目として，IP アドレスとマシン名だけでなく，管理者名，利用者名，設置場所，用途，装置種別，MAC アドレス，備考等の情報を持つこととした．部局の管理者が新規に入力する画面の例を図 2 に示す．一方，部局の管理者が末端利用者からの申請メールを受け取った後に，申請確認する画面の例を図 3 に示す．図 3 の画面の下には，図 2 の画面と同様の入力フィールドが続き，末端利用者が入力した内容が表示される．通常，部局の管理者は図 3 の画面を用い，図 2 の画面を用いることはない．このことによって部局の管理者が入力する作業を減らすことができる．

また，固定 IP アドレスを持たず，DHCP サーバによって動的に IP アドレスが割り振られる場合に対処するために，動的割り当てのフラグの項目を持つこととした．固定 IP アドレスの登録の場合には，この動的割り当てのフラグを立てない．動的割り当てのフラグが立っている場合には，IP アドレスの値は動的割り当てによって取り得る IP アドレスの値の中の一つを示し，取り得る IP アドレスの範囲を備考に記載することとした．

なお，IP アドレスの 2 重発行を避けるために，データベースのレコードの IP アドレスはユニークに限るとした．そのため，動的割り当ての場合に，取り得る IP アドレスの値の一つを入力する場合にもユニークになるように入力する必要がある．もしも，割り当て可能な IP アドレスの範囲の個数を超えてマシン名を登録しなければならない場合には備考欄に記述するなどの対策を必要とするが，実用上問題ないと判断した．

IP アドレスがユニークであるかをチェックするために，IP アドレスを”ABC.DEF.GHI.JKL”のドット表記（A から L は数字）した場合に，DEFGHIJKL の 9 桁の整数値をレコードのキー値として用いた．学内のグローバルアドレスは，多くの大学でクラス B であり，クラス C を他に数個保有していた場合でも，DEF の桁が異なることにより，区別が可能となるためである．もしも DEF の桁すら同一の場合には，DEF を IP アドレスそのものの数字を用いず，区別するための識別数字とすると，キー値として用いることが可能となる．なお，Web ブラウザから操作する時には，キー値の存在は見えず，”ABC.DEF.GHI”のセグメントと JKL の IP ア

ドレスの末尾の数字を指定するインタフェースとなっている．

本学ではセグメントは 24 ビット長であるが，もしもセグメントのビット長が異なる場合でも，その変更は容易に対応することが可能である．なお，クラス B を 1 個クラス C を 3 個持つため，セグメントは 259 個存在している．

4.5 ユーザ管理

ログインアカウントごとに参照することのできるセグメントの範囲は，ユーザ情報テーブルのレコードに 259 個のセグメントごとに参照できるか否かの項目を持たせて実現しているが，この項目はこのシステムの管理者のみが変更することができる．このシステムの管理者は，他人のアカウントのパスワードも上書き方式で強制設定することができる．ログインして利用するユーザ自身がユーザ属性に関して変更ができるのは，自分のパスワードのみである．

なお，パスワードは md5 関数を用いて，キー入力された文字列から変換を行っている．各部局の管理者のログイン用のパスワードがデータベースの中に格納されるが，権限のある集約サーバの管理者がデータベースの中を見たとしてもパスワードは安易にわからないようにしている．

また，末端利用者が各種申請を行なうと部局の管理者へメールが発信されるが，どの管理者へ向けて発信させるかをログインアカウントごとに持たせることにした．メールを受け取った管理者が，自分宛ての申請データのみが見えるようにするため，申請データの中にもその値は保持させ，全部の申請データの中からの絞込みに用いている．

4.6 DNS 設定ファイル

DNS の設定ファイルの前半部分は，セグメントおよびドメインごとにテキストファイルの形で保存されている．CNAME レコードの変更や MX レコードの変更などはこのファイルを Web ブラウザを通して編集する．編集する画面の例を図 4 に示す．

なお，このファイルの 3 行目には SOA レコードの中のシリアル番号が記載されているものとしている．既存ファイルの中身を読み込みその値に 1 加算した値を書き込み直すことを，DNS 登録の更新ボタンが押された時に実行する．もちろん，シリアル番号の最大値 (4,294,967,295) を超えたら 1 に戻るようになっている．この更新ボタンが押されると DNS の設定の後半部分のデータを自動生成し，前半部分と合体した DNS の設定ファイルを更新する．さらに，in.named のプロセスの再起動に相当する”/usr/local/sbin/ndc reload”コマンドを内部で呼び出して自動実行することによって，DNS の設定ファイルの反映を行う．

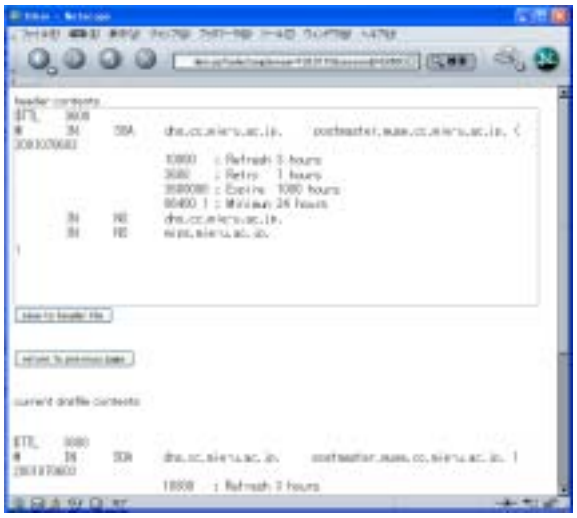

図 4: 編集画面の例

この DNS の設定ファイルやその前半部分のファイルの存在場所は，データベースの中のテーブルで管理されている．

従来忘れがちである SOA レコードのシリアル番号の増加設定は，更新ボタンにより自動インクリメントされるので，前半部分のファイルを手動で直すことはほとんど発生しない．

4.7 バックアップとリカバリ

システムのクラッシュ等に備えるために，データベースのバックアップを取っておく必要がある．それ以上に，システムがさらに改良された全く新しいシステムに入れ替わった場合においても，データの移行がスムーズに行えるように，データベースのデータが，システム依存でなく，CSV 形式のファイルでバックアップできるようにしている．データベースの中の全テーブルのデータについて，テーブルごとに CSV 形式のファイルで出力ができ，かつ，リカバリが行える．

テーブルごとにバックアップとリカバリの操作を

行うためのボタンが，このシステムの管理者がログインした場合のみ画面に現れ，通常のユーザがログインした場合にはそのボタン自身が表示されなくなっている．

4.8 セッション管理

ログイン時にアカウント名とパスワードを入力すると，それ以後は，別ページに遷移してもパスワードを入力し直す必要がないように，セッション管理がされる．JSP では，デフォルトで session 変数が使用でき，この変数に格納されたセッション ID を用いることにする．ログイン画面で生成されたセッション ID を別ページに遷移する時に隠れフィールドを使ってデータ転送し，別ページでは，session 変数の値と転送されたセッション ID の一致を調べる．一致すれば正しくログインしてからそのページに遷移してきたことを意味する．一致しなければ，ログインせずにそのページに直接アクセス要求してきたことを意味し，その場合にはログイン画面からログインするように表示した画面に強制的に遷移させている．

4.9 セキュリティ対策

Apache の設定ファイルにより，Apache のサービスポート番号は，デフォルトとは異なる番号を用いている．また，学外からはアクセスできないように制約を掛けている．

また，Tomcat について，デフォルト設定では Tomcat のサービスポート番号をアクセスすれば Apache 経由でなくても応答するが，設定ファイルにより Apache 経由でないと応答しないように設定している．

また，PostgreSQL について，デフォルト設定ではパスワードなしでデータベースにアクセスできてしまうが，設定ファイルによりデータベースにアクセスするためのパスワード設定を有効にしている．JDBC 経由でプログラムからアクセスする場合も，PSQL コマンドでアクセスする場合もパスワードが必要となるようにしている．

5 実験と評価

本学内で実際に DNS に登録されている実データと情報処理センターで IP アドレス管理されている実データを用いデータベースに登録作業を行った．その結果，2823 件のレコードが登録され，70 個のドメインのデータを一元管理することができた．このデータは，53 台の DNS サーバによって登録されており，それらが 1 台の DNS サーバに集約可能であることを示している．全学の中には，このデータの他に，事務用 LAN などのプライベートアドレスを用いているマシンも存在し，そこで用いられている DNS サーバについては，集約される訳ではない．しかし，グローバルアドレスの 53 台の DNS サーバを 1 台に集約できることで十分な効果が期待できる．

SOA レコードのシリアル番号の記述位置を 3 行目と固定にしたが，実際の設定ファイルでは先頭に $TTL 3600　と書かれた場合には 3 行目に位置し，書かれていない場合は 2 行目に位置していたため，3 行目に統一することは問題を生じなかった．

また，動的割り当て箇所については，割り当て可能な IP アドレスの範囲の個数を超えてマシン名を登録しなければならない場合は存在せず，問題を生じなかった．

登録時に問題となったのは，2 つの A レコードが同じ IP アドレスを持って DNS 登録されている場合が 2 箇所存在し，それらは，データベースの中の IP アドレスをユニークキーに限定しているために，そのままでは登録ができなかったことである．しかし，DNS の設定ファイルの前半部分へ重なった A レコードの一つを移動させることで解決を行った．なお，A レコードでなく CNAME レコードに記述を変更する解決方法もあったが，既存の記述をそのまま変更しないで記述した．

初期導入時の作業負担としては，bind の導入まではほとんど同じであり，その後，Tomcat,Apache,PostgreSQL,J2SE などのミドルウェアのインストールに時間を要した．これに関しては Solaris の場合には，バイナリ提供されていない場合が多いが，Linux の場合には，バイナリ版が提供されているので，簡単にインストールすることが可能である．

次に，著者の所属する教育学部において，末端利用者からの申請機能を除いた本システムの実運用を開始した．運用して 1 年弱であるが大きなトラブ

ルは起きていない．末端利用者からの申請機能については最近開発したため実運用はこれからである．運用して注意を要したのは，データベース中の日本語のデータが字化けしてしまうことである．これはTomcat を起動する時に環境変数 LANG が ja に設定されていないと発生する．このため，マシン起動時に Tomcat をシェルスクリプトで自動起動させ，その中に LNAG を ja に設定する行も埋め込むことで解決させている．

運用を継続させながら評価すべきこととしては，集約 DNS サーバの運用管理の負担度合いと，セグメントやドメイン名の変更があった場合の変更のしやすさがある．まず，集約 DNS サーバは，従来の DNS サーバと比べて，インストールすべきミドルソフトウェアが増加する．そのため，セキュリティホールが発見された時にパッチ当てを作業しなければならないことが増えると考えられる．しかし，これは集約サーバ１台のみのパッチ当て作業であり，53 台の DNS サーバの各管理者がパッチ当て作業をしていた従来のトータル時間と比べると大幅に削減すると考えられる．

次に，セグメントやドメイン名の変更があった場合は，セグメントとドメイン名の対のテーブルと設定ファイルの存在場所を格納するテーブルの中のレコードの値を変更する必要がある．これに対しては，バックアップ＆リカバリの節で述べたようにテーブルの中身が CSV 形式のファイルで出力できるので，そのファイルをテキストエディタで編集してからリカバリすれば，ドメイン名の変更など容易に行うことができる．ドメインを追加する場合には，DNS 設定ファイルの前半部分に追加するドメインに対する記述を書く必要があるが，これは従来と全く同じように設定ファイルを編集することと等価なので，作業は難しいことではない．

今後，運用の適用範囲を広げていった場合には，集約 DNS サーバの DNS 問い合わせの応答速度の問題が考えられる．全学の DNS 問い合わせを引き受けることになるが，学内のネットワークはギガビットネットワークに高速化され，ネットワーク性能としては問題がない．サーバの性能としても，経験上問題はないと思われる．むしろ，マシンが故障した場合に対処するために，全学向け実運用に入った場合には DNS のセカンダリネームサーバを用意する必要がある．もしも，端末教室のようなアクセスが集中しやすい箇所に対し，応答速度に不安がある場合には，安価な Linux マイクロサーバ [7] を DNS のセカンダリネームサーバとして設置することが考えられる．プライマリネームサーバでなくセカンダリネームサーバとしておけば，導入後の管理作業が少なくて済むからである．

6　まとめ

セキュリティ対策のための大学内一括管理向け DNS 登録システムを開発した．このシステムは，(1)DNS 登録を１台のマシン上でありながらも複数の管理者が各自の管理領域のみ扱うことができ，従来の分散管理方式と同様の負担で運営することができる．(2)DNS 登録せずに IP アドレスを割り当てたマシンの管理も同時に行うことができ，従来の２重管理をなくすることができる．(3) マシンの情報は末端管理者がデータベースに入力する形態を持ち，部局の管理者によるデータ投入作業を軽減することができる．(4) 学内の全マシンの情報がデータベースの中に登録されるために，情報処理センターのような対外接続管理部局において，セキュリティ上必要となる学内全マシンの情報一元管理が可能となる．以上の４つが可能となるシステムである．

この結果，(1) 発生費用を抑制することができる．(2) パッチ当て作業など全管理者の作業時間を抑制することができる．(3) セキュリティ対策を推進させることができる．以上の３つの利点を生むことができる．

全学で実運用されれば，本学の場合で 53 台の DNS サーバが１台に集約でき，セカンダリネームサーバを用意するなどの安定稼動のための配慮を加えれば，十分に効果を達成し運用可能であると思われる．

今後は，このようにデータベースの中に学内全マシンの情報が投入されてくると，ますます応用が広がると考えられる．例えば，ファイアウォールを通過させるか否かのフラグをレコード項目に追加すれば，ファイルウォールの通過申請管理データベースとしても活用ができるようになると期待できる．

参考文献

[1] 関谷勇司, 石原知洋:”DNS のセキュリティ対策と運用状況の調査ツール”，情報処理学会誌,Vol.41,No.12,pp.1373-1379 (Dec 2000)

[2] HDE Controller: (株) ホライズン・ディジタル・エンタープライズ製, http://www.hde.co.jp/controller/solaris/ (2003 年 7 月現在)

[3] 坂本江見, 高井正三:”WWW とデータベースの連携によるネットワーク運用システムの開発”, 学術情報処理研究,No.1,1997,pp.55-57 (Oct 1997)

[4] 松原義継, 只木進一:”Web ブラウザを用いた DNS 管理システムの開発”, 情報処理学分散システム／インターネット運用技術研究報告,2001-DSM-21,pp.31-36 (May 2001)

[5] 江藤博文, 只木進一:”WEB ベースネットワーク運用システムの開発”, 情報処理学分散システム／インターネット運用技術研究報告,2001-DSM-24,pp.25-30 (Nov 2001)

[6] Paul Albitz,Cricket Liu(著), 高田, 小島, 小舘 (訳): ”DNS & BIND 第 4 版”,O’REILLY (Feb.2002)

[7] OpenBlocks:ぷらっとホーム (株) 製, http://openlab.plathome.co.jp/OpenBlockS/dns.html (2003 年 7 月現在)