マルチメディア通信と分散処理 85－27
（1 9 9 7. 1 1. 7）

インタネットにおける管理簡易化－ DNS と BGP 管理

鍾 順吉[†] 門林 雄基[‡] 山口 英[†] 尾家 祐二[§]
cheng@ai3.net *youki@center.osaka-u.ac.jp* *suguru@is.aist-nara.ac.jp* *oie@cse.kyutech.ac.jp*
[†]奈良先端科学技術大学院大学情報科学研究科
[‡]大阪大学大型計算機センタ
[§]九州工業大学情報工学部

インターネット関連技術の急速な技術革新は、常に新しい技術的知識や運用ノウハウを必要とする。このため、インターネット利用者人口のごく少数しか占めないネットワーク管理者やサーバ管理者の管理負担はますます大きくなる一方である。また、ネットワークの広がりの中で、ネットワーク管理者はますます不足しつつある。このような問題を解決する手法として、管理者人口の増加の妨げの原因と考えられる複雑なインターネットのネットワーク構築技術やサーバ技術の深い知識・経験を必要としないインターネットの零知識管理技術を提案する。本稿は特に DNS や BGP 管理について、この技術の有効性を検証する。

Toward Zero Internet Administration Technology – The case of DNS and BGP Administration

Cheng Soon Giap[†] Youki Kadobayashi[‡] Suguru Yamaguchi[†] Yuji Oie[§]
cheng@ai3.net *youki@center.osaka-u.ac.jp* *suguru@is.aist-nara.ac.jp* *oie@cse.kyutech.ac.jp*
[†]Graduate School of Information Science, Nara Institute of Science and Technology
[‡]Computation Center, Osaka University
[§]Faculty of Computer Science & System Engineering, Kyushu Institute of Technology

The community of knowledgeable network engineering practitioners is very small. In this paper, we discuss Zero Internet Administration Technology which is one of the most important technology for today's network system. This technology significantly reduce the job of network administrator in particular and computer user in general by eliminating repetitive tasks, application configuration job and etc. We discuss the technology in the case of Domain Name System(DNS) and inter-domain routing auto-configuration in BGP. In both cases, we present our solutions by implementing integrated tools with GUI for managing it.

1 Introduction

In order to manage today's system effectively and to plan intelligently for the future use of the network systems, network manager needs an understanding of the network management technology and also a thorough grasp of the details of the existing and evolving standard. However, the community of uptodate individuals is very small. Network Administrators will be challenged with the amount of new information they have to learn in order to keep up with these new technologies.

Practical, detailed network information we have today, which are collected through years of practical experience, is frequently useless unless it can be put together in a form of uniform utilities and shared with others who need it. We strongly feel that the main goal of Zero Internet Administration Technology is to transform all the "hand-to-mouth" style of management practice into easy-to-use application.

In this paper we focus on practical usage of the technology in the case of Domain Name Sever(DNS) and inter-domain routing configuration of Border Gateway Protocol(BGP). However, our work with Zero Internet Administration Technology is not limited to these two fields. Our goal is to develop from this experiences a strong techniques that will contribute to the development and acceptance of any other application. Our long-term goal is to create a stable network infrastructure whose reliability and throughput at various levels will be secured with a minimum configuration management.

2 Case Studies: Domain Name System(DNS)

The goal of domain names is to provide a mechanism for naming resources in such a way that the names are usable in different hosts, networks, protocol families, internets, and administrative organizations [8].

The implementation of DNS used on most UNIX systems is the *Berkeley Internet Name Domain* (BIND) software. DNS name service software is conceptually divided into two components – a resolver and a name server. The *resolver* is the software that forms the query and the name server is a network service that enables clients(*resolver*) to name resources or objects and share this information with other objects in the network.

Users can configure the system to use BIND as a replacement to the older host table lookup of information in the network hosts file*/etc/hosts* [7]. However, in order to configure a machine to function as a name server, we need to know a lot of BIND syntaxs. In addition, it is not only one file we need to modify but it include multiple files. One file maps all the host names to addresses, one file map the addresses back to host names(reverse mapping) and etc.

2.1 Design of a System for managing DNS

2.1.1 Task automation

Whenever you rename a host name or install a new machine into your network, the database files in DNS must be modified. However, the syntax of the DNS files lends itself to making mistakes.

Failing to increment the serial number is the most common mistake made when updating a name server. Every time you make a change to a name server database file, you must increment its serial number. Only by doing this will secondary servers know they need to reach into your system and make a zone transfrer.

Network administrators could easily spend every minute of every day performing all these periodic maintenance tasks unless they find a method of automation.

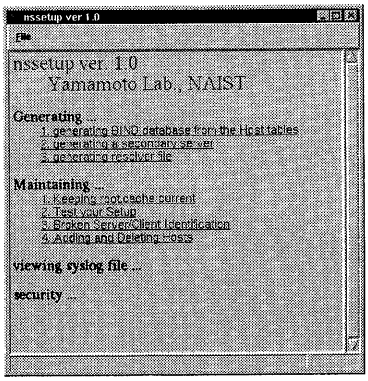

nssetup ver 1.0
File
nssetup ver. 1.0
Yamamoto Lab., NAIST
Generating ...
1. generating BIND database from the Host tables
2. generating a secondary server
3. generating resolver file
Maintaining ...
1. Keeping rootcache current
2. Test your Setup
3. Broken Server/Client Identification
4. Adding and Deleting Hosts
viewing syslog file ...
security ...

Figure 1: nssetup

Tasks that are repeated often and are complicated or tedious are good candidates for both simplification and automation [9].

3 Our Implementation: *nssetup*

Various utilities for configuring BIND already exist. However, most of them are either not integrated or difficult to use. Since multiple tools are required to perform different tasks, it is not an easy way for a beginner to know which utility is relevant. To overcome these limitations, we have developed *nssetup*, a *tcl/tk*-base [4] tool which is meant to be useful, simple and integrated. Figure 1 shows the main user interface of *nssetup*, where a user can select the task he would like to perform from it.

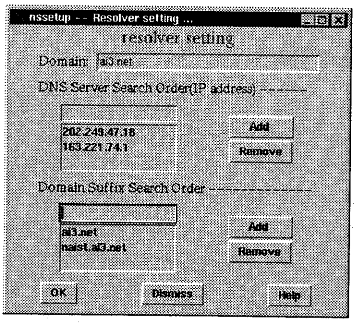
3.1 Functions

3.1.1 Resolver Configuration

nssetup provide an interface as shown in Figure 2 to handle the resolver configuration. We try to implement the interface to have almost similar outlook as provided by Window95 so that user moving from Microsoft OS to Unix will not be reluctant to use the interface.

3.1.2 Configuring Name Server using host table

While the resolver configuration requires one configuration file, several files are used to configure name server as stated above. As a result, keeping consistency of all the files is very important. Without a utility to help automated this process, not only a lot of time have to be spent to input all



nssetup -- Resolver setting ...

resolver setting

Domain: ai3.net

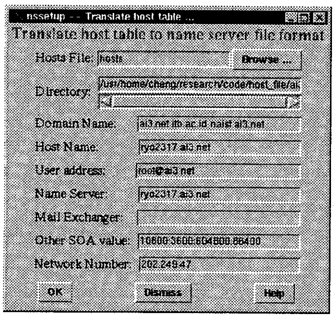
DNS Server Search Order(IP address)

202.249.47.18
163.221.74.1

Domain Suffix Search Order

ai3.net
naist.ai3.net

Figure 2: Resolver Configuration



nssetup -- Translate host table ...

Translate host table to name server file format

Hosts File: hosts

Directory: /usr/home/cheng/research/code/host_file/ai

Domain Name: ai3.net itb.ac.id naist.ai3.net

Host Name: ryo2317.ai3.net

User address: root@ai3.net

Name Server: ryo2317.ai3.net

Mail Exchanger:

Other SOA value: 10800:3600:604800:86400

Network Number: 202.249.47

Figure 3: Translate host table into name server file format

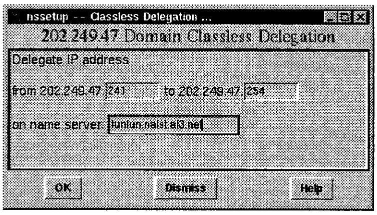
the necessary data but also a lot of care have to be taken on the BIND database file syntax.

Realizing that setting up BIND is almost well-structured process for converting host table information into name server information, *nssetup* provide some high-level tools such as generating DNS database of an entire zone from an */etc/hosts* file.

In contrast to command line application, given the */etc/hosts* file, *nssetup* will automatically find an appropriate name of your domain and your network numbers as shown in Figure 3; nevertheless you will be able to correct it if you think that a better one is available.

3.1.3 Keep the cache uptodate

Each server maintains a cache initialization file, which contains the information needed to begin building a cache of domain data when the name server starts. It changes over the time. Keeping the *root.cache* file more uptodate than the one that comes with the BIND distribution is indispensable. *nssetup* is able to ftp the file from the internet automatically to keep the cache initialization file up to date.



nssetup -- Classless Delegation ...

202.249.47 Domain Classless Delegation

Delegate IP address

from 202.249.47. 241 to 202.249.47. 254

on name server: lunlun.naist.ai3.net

Figure 4: Classless in-addr.arpa delegation

```
...
70   IN   PTR     ait-iptv.ai3.net.
;
subnet241 86400   IN   NS     lunlun.naist.ai3.net.
241  IN  CNAME  241.subnet241.47.249.202.IN-ADDR.ARPA.
242  IN  CNAME  242.subnet241.47.249.202.IN-ADDR.ARPA.
243  IN  CNAME  243.subnet241.47.249.202.IN-ADDR.ARPA.
...
252  IN  CNAME  252.subnet241.47.249.202.IN-ADDR.ARPA.
253  IN  CNAME  253.subnet241.47.249.202.IN-ADDR.ARPA.
254  IN  CNAME  254.subnet241.47.249.202.IN-ADDR.ARPA.
...
```

Figure 5: This file, db.202.202.249 was generated using the classless delegation utility of nssetup

3.1.4 Utility for updating the database

It is easy to add or deleting the name server database files in *nssetup*. The serial number will be updated automatically. Without this function, a user have to keep on finding out every domain and network file for which the host have an address. Since these data are in separate files, it is easy to err.

3.1.5 Classless in-addr.arpa delegation support

One of the problems encountered when assigning address space on non-octet boundaries is that it seems impossible for such an organization to maintain its own reverse ("in-addr.arpa") zone autonomously. An Internet-Draft, [3] describe a reverse delegation method which solved the problem of assigning variable prefixes to unrelated organization. However, the concept is not easy for most of the user to understand.

By providing 3 necessary argument to the interface shown in Figure 4 , *nssetup* will generate the necessary fields in the related file as shown in Figure 5.

4 Evaluation on *nssetup*

An experiment had been conducted in our lab to evaluate *nssetup*. The experiment requires participant to setup a name server for a domain consists of about 10 hosts. A host table had been provided

ID	Time (minutes)	
	1	2
without any tool	600	120
using a command line tool	30	8
using *nssetup*	3	2

Table 2: Time use for configuring a name server

for generating the domain database. First of all, time was measured without any tool to help them. Following the first test, participant began to use a command line tool, *h2n* [6] to perform the same task. Finally *nssetup* was used for the same purpose.

There are 2 computer users participated in the experiment. The participant's experience with computers was distributed in terms of their experience in managing a network system. Table 1 provides a brief description of each participant.

The total time taken to complete each task is shown in Table 2. Our results show that the total time used for configuring a name server without using any utility is very long comparing to the other two approaches. The total time an experience user used for configuring a name server is about 40 times a user without any real experience in managing a network system. This is not surprising in view of the fact that a lot of typing and reference to technical documents on configuring BIND are required to perform all the tasks. To configure a single domain name server, we need at least 4 files to configure and also have to ftp one from the internet. In addition, typing mistake and syntax error occurred very often and it take times to debug it.The time are expected to be shorter with consistent practice or repetition.

Another remarkable from the table is that despite of the experience of participantes, they took very short time to configure their server by using *nssetup*. Although utility like *h2n*, which is designed only for performing this task, also provide the same function to automate the configuration of name server, *nssetup* provide additional GUI to guide the user in using necessary arguments. In addition, most of the arguments are provided automatically in the interface. *nssetup* proved to be useful for both experience and non-experience user.

5 Case Studies: BGP Routing Configuration

The Net has been maintained by people with professional skill. The troubles in the networks often require experience and therefore, takes time in finding and fixing them. Routing problems were frequently caused by router misconfiguration. Careless configuration of a router can disrupt the normal operation of a network. The increasing size of networks makes their management more and more difficult.

The Border Gateway Protocol(BGP), defined in RFC 1771 [10], provides inter-domain routing between autonomous systems. In this protocol, the network address to announce, value of the BGP timer, filter-list and IP addresses of their routers which to establish a peer relationship with are configured. Configuration for each router has a lot to do with routing. They play an important role when the routers make routing decisions. These configuration includes interface information, what routing protocol to run, whether to redistribute routing information from one routing process to the other, which network prefix to announce and so on. So a change or mistake in the configuration can have a big impact on the network. It can lead to non-optimal routing, it can lead to certain destinations becoming unreachable, or can even partition the Internet [1].

6 Our Suggestion: *freeRoute*

Our work concentrates on BGP-4 and how GUI techniques can help users in configuring their routers. The work were done on Gated [5] routers. Modules for other routers are left for future work. We are developing a tool called *freeRoute* to help the task of configuring a router.

6.1 System

The design of *freeRoute* was motivated by our analysis of existing systems. We have identified the following key issue that should be addressed when configuring a router.

- Understanding how a large network system's router interact is the key in maintaining and configuring a router. GUI will enable human manager to visualize the internetwork. Visualizing ideas, especially when they're presented as colorful, appealing graphics is useful in providing effective tools for tackling the

ID	Computer Use	Position	Background
1	work/research	Master Course 1 year Student in Computer Science	Extensive use of PC and Workstation
2	work/research	Master Course 2 year Student in Computer Science	Experience in network management

Table 1: participants background

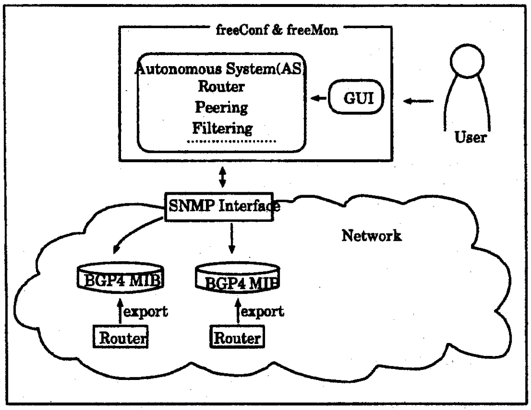


Figure 6: *freeRoute*: Architecture

problems. Making the unseen visible is important in routing configuration.

Configuring routers to realize various policies is not an easy task. *freeRoute* try provide a full GUI for network administrators to draw their network topology and input all the necessary routing policy inside it. We believe this way will ensure them having a clearer image of their network topology. The architecture of *freeRoute* is shown in Figure 6.

As shown in the figure, *freeRoute* consists of several parts:

- SNMP interface
  We use SNMP, an application-layer protocol designed to facilitate the exchange of management information between network devices, to gather information. SNMP has the advantage in absorbing the differences between the vendors. The largest advantage of using SNMP is that its design is simple, hence it is easy to implement on a large network, for it neither takes a long time to set up nor poses a lot of stress on the network. The net result of this simplicity is network manager that is easy to implement and not too stressful on an existing network. SNMP helped *freeRoute* in finding neighbor router which is involves in routing policy.

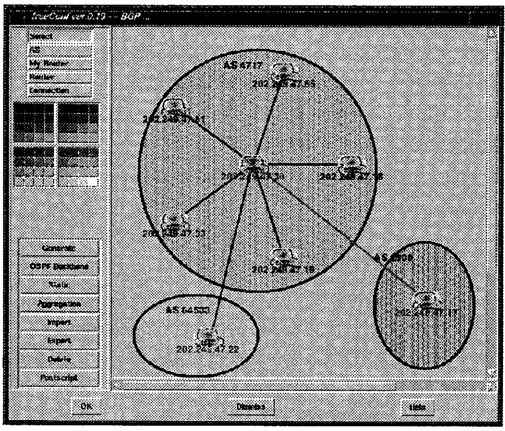


Figure 7: *freeConf*: Routing Configuration

- *freeMon*
  This is the Network Monitor Manager of *freeRoute*. The data provided by *freeMon* provide the information on the condition of the network. Constant monitoring of these statistics will help in detecting possible troubles.
- *freeConf*
  This tool provide the necessary utility to draw the topology network of the router to be configured. Using the drawn map, and some necessary data required, configuration files will be generated automatically. Figure 7 shows the main user interface of *freeConf*, where a user can select the task he would like to perform from it.

freeConf consists of:

- Autonomous System(AS) Object
  An AS number will be associted with each AS object. It describe the policies of the AS. Policies for importing routes and exporting routes will be specified whenever an AS object is created.
- Router Object
  Since we are configuring for one of the many routers in the Internet, the router in *freeRoute* are divided into 2 categories: ***myRouter*** which is the router to be configured and ***Router*** which is other router in the Internet which is

related to our router to be configured.

- Peering(link)
  A line will be drawn between 2 routers that are having routing information exchange.

At the moment of this writing, *freeConf* of *freeRoute* is almost complete for BGP configuration. However, *freeMon* is still to be implemented. We are trying to collect most of the needed functions for *freeMon* to function as expected.

Although it is easy to express the topology of the network into GUI, we found that it is difficult to visualize the ideas of importing and exporting routes. Morever, it is also a problem when neighboring ASs becoming too many to express. *freeRoute* is able to express the topology but expressing a topology that consists of hundreds or thousands of ASs is still a problem. We are studying if 3 dimensions expression is needed in *freeRoute* to show a clear topology of multiples ASs.

7 Future work

Although our Zero Internet Administration Technology is potentially beneficial, it is not particularly easy to achieve. Because there are too many fields that need this technology, we are now concentrating on 3 major applications. First, it is DNS which is described in section 2 and [2]. Second, we are now interesting in automating the process of internet routing configuration and trouble shooting, which is discussed section 6 in this paper. And finally, we are also interested in managing the configuration of *sendmail* program, which is the daemon responsible for delivering electronic mail.

Our next step is to perform user studies and evaluations in real operational environments.

8 Summary and Conclusions

In this paper, our objective was to clarify the importance and benefits of our Zero Internet Administration Technology. We explained the idea in the case of DNS and routing configuration.

Although it is not an easy task to transform all the hand-to-mouth style of knowledge into useful application that supports all aspects of a task well, the effort required to make it a reality is indispensable.

nssetup and *freeRoute* are undoubtedly our foundation for integrating other concept into Zero Internet Administration Technology.

Acknowledgements

We would like to thank NAIST's Division of Information Network members for their useful suggestions and helpful criticisms.

References

[1] Cengiz Alaettinoglu, "Scalable Router Configuration for the Internet," *http://www.isi.edu/cengiz/publications/*, IEEE IC3N '96. October 1996.

[2] Cheng Soon Giap, Youki Kadobayashi and Suguru Yamaguchi, " Zero Internet Administration Approach: the case of DNS," The 12th International Conference on Information Networking (ICOIN-12) (to appear).

[3] Havard Eidnes and Paul Vixie, "Classless IN-ADDR.ARPA delegation," *draft-ietf-dnsind-classless-inaddr*, April 1997.

[4] John K.Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

[5] Merit GateDaemon Consortium, *http://www.gated.merit.edu/*.

[6] Paul Albitz and Cricket Liu, *DNS and BIND*, O'Reilly and Associates, Inc, 1997.

[7] Paul Vixie, "Name Server Operations Guide for BIND, Release 4.9.6," Internet Software Consortium La Honda, CA.

[8] P.Mockapetris, "Domain Names - Implementation and Specification," *rfc1035*, November 1987.

[9] Unix System Administration Independent Learning(USAIL) Project, "Automating Tasks," *http://www.uwsg.indiana.edu/usail/index/automate.html*, 1996.

[10] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," Request for Comment RFC 1771, March 1995.