Powered by DNS

김승영

nobreak@openbird.com

Powered by DNS

지은이 김승영

3rd Edition, 최종수정일 **2001**년 **12**월 **16**일**, Revision 3.4.5**

HTML(Divided) (./) HTML(Combined) (PoweredByDNS.html) SGML(DocBook) (PoweredByDNS-3.4.5.tar.gz)

잘못 쓰여진 내용이 있다면 약간의 시간을 할애하여 저자에게 알려주시기 바랍니다**.** 이것은 다음 사람들에게 올바른 지식을 전달하는데 큰 도움이 됩니다**.** 본 글의 최종 수정본은 **http://bsd.openbird.com/doc/PoweredByDNS/**에서 구할 수 있으며**,** 도움이 된다면 자유롭게 활용하셔도 좋습니다**.**

Table of Contents

List of Figures

1Chapter. Domain Name System의 이해

Domain Name System이란 이름과 IP 주소를 매핑하여주는 거대한 분산 네이밍 시스템이다. 인터넷에서 사용되는 IP(Internet Protocol), 그리고 IP의 상위에서 동작하는 넷스케이프 같은 응용들은 210.105.79.103 과 같이 표현되는 IP 주소만을 인식하게 되는데, 이러한 IP 주소는 기계입장에선 해석하기 수월하지만 기억하기가 어렵고, IP 주소만으로는 서비스 유형을 예측하기 힘들다는 단점이 있다. 인터넷의 도입 시절인 ARPANET 시절부터 IP Address를 이름으로 명명하여 사용하고자 하는 노력이 시도되었고, 많은 시행착오는 지금의 DNS 메커니즘으로 발전하였다.

ARPANET 시절에는 호스트의 수가 많지 않았기에 NIC(Network Information Center)으로부터 일정 주기마다 호스트 명단 파일(HOSTS.TXT)을 받아 /etc/hosts에 저장하여 사용하였다. 그러나 점차 인터넷의 규모와 호스트 수가 증가함에 따라 새로운 이름 명명 체제의 필요성이 대두되었고, 1983년 Paul Mockapetris가 RFC882 (ftp://ftp.openbird.com/pub/rfc/rfc822.txt), RFC883 (ftp://ftp.openbird.com/pub/rfc/rfc883.txt)(현재는 RFC1034 (ftp://ftp.openbird.com/pub/rfc/rfc1034.txt)로 대체됨)에 새로운 명명 체제에 대한 구현을 공식 발표하며, 크게 네임스페이스의 계층 구조, 분산 데이터베이스, Email 라우팅 개선을 주안점으로 DNS가 탄생하였다.

1.1. DNS Namespace

ARPANET의 중앙 관리 체제에서는 하나의 파일로 모든 호스트들을 관리하였지만, DNS에서는 이것을 각 도메인 별로 트리화 하여 Figure 1-1과 같은 형태로 관리한다.

Figure 1-1. Namespace 예제

디렉토리 구조와 유사함을 알 수 있는데, Root domain(도트로 표시되는)은 Top level 도메인에 관한 정보를, Top level 도메인은 그 하위 도메인에 관한 정보를 유지/관리하는 구조를 취한다. 이러한 정보의 계층구조로 인하여 정보는 각 도메인의 네임서버(NS:Name Server)로 분산, 관리된다.

예로 YAHOO.COM 도메인은 COM 네임서버에 등록되어 있고, WWW.YAHOO.COM은 YAHOO.COM 네임서버에 등록, 관리된다. 따라서 AV.YAHOO.COM을 등록하기 위해서는 YAHOO.COM 도메인을 관리하는 네임서버의 관련 레코드만을 수정함으로써 가능하다. 이러한 위임구조는 증가하는 인터넷 호스트에 대한 관리를 효율적으로 가능하게 해준다.

1.2. Domain Name Resolving

Figure 1-2. 도메인 네임 **Resolving**

통신을 위한 TCP/IP 패킷엔 도메인명을 위한 공간이 없다. 따라서 도메인명에 대한 IP 변환 작업(Resolving)을 선행하게 되는데, Figure 1-2는 이러한 Resolving 과정을 보여준다.

1. Client상의 응용이 WWW.YAHOO.COM 에 접속하기 위해 자신의 Local Name Server(TCP/IP 설정시 명시한)에 질의한다.
2. Local NS는 먼저 자신의 캐쉬에 자료가 있는지 확인한 후 발견되지 않을시 Root NS(Root NS의 목록은 갖고있다)에 질의를 던진다. 그러나 Root NS도 WWW.YAHOO.COM 의 자료를 갖고 있지 않으므로, COM 도메인을 관리하는 NS를 참고하라는 답변을 보내준다.
3. Local NS는 다시 COM NS에 질의를 던지고, COM NS는 다시 YAHOO.COM의 NS를 일러준다. (루트(도트)와 COM 도메인은 Root NS에서 같이 관리되기 때문에 실제로 본 과정은 일어나지 않고 (2)번에서 바로 YAHOO.COM NS를 참고하라는 답변이 나온다.)
4. Local NS는 YAHOO.COM NS에 질의한다. YAHOO.COM NS는 서브도메인에 대한 자료를 관리하는 실제 NS 이므로, WWW.YAHOO.COM 에 대한 IP 204.71.200.74를 답변(authoritative answer) 한다. (실제 WWW.YAHOO.COM 에 대한 IP를 resolving 하면 매번 다른 IP 주소가 넘어오는 것을 볼 수 있는데, 이는 사이트가 다수의 미러링 서버로 운영되고, 부하 공유를 위해 해당 도메인에 다수의 IP를 매핑하였기 때문이다. (참고: 미러링 서버간의 부하 공유)
5. 마지막으로, Local NS는 Client에게 결과를 전송한다.

1.3. 도메인명에 사용가능한 문자셋

참고: RFC1035 (ftp://ftp.openbird.com/pub/rfc/rfc1035.txt) p7, RFC1032 (ftp://ftp.openbird.com/pub/rfc/rfc1032.txt) p2

’A-Z’, ’a-z’, ’0-9’, ’-’ 가 사용될 수 있다. 도메인명은 반드시 숫자나 문자로 시작하여야 하며, 전부 숫자여서는 안 된다. (초기 [RFC1035 (ftp://ftp.openbird.com/pub/rfc/rfc1035.txt)]에서는 도메인명이 반드시 문자로 시작하도록 제한하였지만, [RFC1123 (ftp://ftp.openbird.com/pub/rfc/rfc1123.txt) p12]에서 완화되어 3com.com 같은 도메인 사용이 허락되었다.) 또한 하이픈이 앞/뒤에 사용될 수 없고, 도트로 분리되는 각 문자열(Each segment)은 최고 63 단어(octet)까지 사용할 수 있도록 구현되어 있으며(NIC에서는 12단어 이하로 사용하기를 권장), 대/소문자는 구분되지 않는다.

덧붙여, 도메인명은 등록기관(kr 도메인의 경우 KRNIC (http://www.nic.or.kr/), com/net/org 등의 도메인은 Network Solutions (http://www.networksolutions.com)을 대표로 ICANN (http://www.icann.org)의 심사를 획득한 등록 대행 업체들)에 따라, 조금씩 상이한 규칙이 적용됨을 유의하자. KRNIC은 숫자로 시작하는 도메인명을 허용하지 않는 반면 세그먼트를 63자까지 풀어주지만, Network Solutions에서는 숫자로 시작하는 도메인을 허용하고 세그먼트를 22자로 제한한다.

참고로 도메인명은 통상적으로 Email에도 사용되므로 RFC822 (ftp://ftp.openbird.com/pub/rfc/rfc822.txt) 규칙도 준수해야 할 필요가 있음을 알아야 한다. 하지만, 이는 충분히 자유로워 앞의 규칙을 포용하므로, 크게 신경쓸 필요는 없겠다.

(제작되는 소프트웨어들은 표준의 규격보다 확장되고 유연하게 구현되어 실제 숫자로만 된 도메인(예:411.com)도 존재한다. 하지만 표준에 위배되는 것은 모든 응용에서 통용되리라 기대할 수 없다는 말이기도 하다)

2Chapter. Name Server 운영

2.1. Name Server 유형

네임서버는 Primary, Secondary, Cache only server로 구분된다.

Primary server는 해당 도메인을 관리하는 주 네임서버이고, Secondary server는 특정 도메인에 대한 back-up copy를 유지하는 서버이다. Secondary는 Primary가 비정상 운행될 때와 부하를 분산시키기 위해 운용하며, 다수가 존재할 수 있다.

보통 도메인을 관리하기 위해서는 Primary, Secondary 서버가 필요하게 되며, Secondary는 원칙적으론 외부 네트웍에 위치시켜 정전 등의 사태로 Primary가 다운되었을 때를 대비한다. 따라서, 도메인을 운영하기 위해서는 최소 2대(Primary * 1, Secondary * n) 이상의 네임서버가 요구된다.(기술적으로 Resolver의 입장에서는 Primary와 Secondary가 구분되지 않기에 Primary 만으로도 운영은 가능하나 권고되진 않는다)

Cache only server는 도메인에 대한 데이터를 관리하지는 않고, resolving만을 처리해 준다. 만약, 본사와 지사가 있고 이 회사의 Primary, Secondary Name server가 모두 본사에 위치한다고 할 때, 지사에 위치한 네트워크 유저들은 Local DNS server가 없게 된다. 이럴 경우 도메인 resolving이 요구될 때마다 다른 네트워크(본사)로 접속을 시도하게 되므로 약간의 딜레이가 생기게 되며, 본사 네트워크가 단절 되었을시 지사도 실질적으로 인터넷 사용이 불가능한 단점이 있다. 이럴 때 지사에 Cache only server를 운용하면 효과적으로 문제를 해결할 수 있다.

2.2. BIND(Berkeley Internet Name Daemon) 설치

Name server를 운용하기 위해서는 서버측 데몬 프로그램이 필요하게 되는데, 이중 BIND는 db 파일의 구성이 손쉽고 표준을 충실히 따른 검증된 도구로서 인터넷에서 가장 널리 사용된다. 대부분의 Unix 시스템에서는 BIND가 이미 설치되어 있다. `/usr/sbin` 디렉토리에 **in.named** 혹은 **named**가 존재함을 확인하고, BIND가 이미 설치되어 있을 경우에는 다음과 같이 설치된 BIND의 버전을 확인한다. (BIND가 동작중이여야 함)

```
$ dig @ns.nobreak.com txt chaos version.bind. | grep VERSION
VERSION.BIND.            0S CHAOS TXT    "8.2"
```

배포처인 ISC(Internet Software Consortium) (http://www.isc.org) 에서 BIND의 최신버전을 확인하고, 버전차이가 많거나 현재 버전에 심각한 문제가 보고되었다면, 업그레이드를 고려하고, BIND-4가 설치되어 있다면, BIND-8로 마이그레이션하여 새로운 흐름에 조인하는것도 나쁘지 않겠다.

BIND의 설치는 매우 간단하다. ISC FTP사이트 (ftp://ftp.isc.org/isc/bind/src/)에서 최신 버전의 소스를 내려받아, 압축을 푼후 다음과 같은 명령을 입력하는 것이 설치에 필요한 전부이다.

```
# make clean depend all install
```

그리고, 시스템 rc 스크립트를 적절히 수정하여 시스템 부팅시 BIND가 자동으로 구동될 수 있도록 한다. (FreeBSD: `/etc/rc.conf`, `/etc/rc.network`, Solaris: `/etc/init.d/inetsvc`, Linux: `/etc/rc.d/init.d/named`)

Windows NT, OS/2, MacOS 등에서 BIND를 운용하고자 한다면, 다음 페이지를 참고하자.

http://www.dns.net/dnsrd/docs/exotic.html

2.3. 퍼블릭 도메인(Public Domain) 신청

Primary, Secondary 네임서버가 준비되었고 신청할 도메인이 결정되었다면, 상위 도메인 관리 기관(kr 도메인의 경우 KRNIC (http://www.nic.or.kr/), com/net/org 등의 도메인은 Network Solutions (http://www.networksolutions.com)을 대표로 ICANN (http://www.icann.org)의 심사를 획득한 등록 대행 업체들)에 도메인을 신청하여 발급(네임스페이스상에 링크) 받게 된다. 도메인 신청양식은 기관마다 조금씩 상이하지만 일반적으로 사용기관, 책임자, 관리자, 결제자, 네임서버 정보가 요구된다. 이중 신청 도메인을 네임스페이스에 링크하기 위한 네임서버 정보는 다음과 같이 작성한다.

```
2.   Complete Domain Name.......: NOBREAK.COM
7a.  Primary Server Hostname....: NS.NOBREAK.COM
7b.  Primary Server Netaddress..: 210.105.79.2
8a.  Secondary Server Hostname..: NS2.NOBREAK.COM
8b.  Secondary Server Netaddress: 210.105.79.3
```

"NOBREAK.COM"이 등록되었다는 메시지를 받았다면, 다음과 같이 해당 도메인의 등록 여부를 확인한다.

```
$ nslookup -type=ns nobreak.com
Server:  ns.nobreak.com
Address:  0.0.0.0

nobreak.com      nameserver = ns.nobreak.com
nobreak.com      nameserver = ns2.nobreak.com
ns.nobreak.com   internet address = 210.105.79.2
ns2.nobreak.com internet address = 210.105.79.3
```

해당 도메인에 대한 네임서버가 신청한 것과 같이 표시된다면, 등록이 바르게 진행된 것이다. 아직 등록이 안되었다면, 다음과 같은 메시지를 볼 수 있다.

```
*** local.name.server can't find nobreak.com.: Non-existent host/domain
```

"도메인 NOBREAK.COM을 신청하는데 어떻게 그 하부에 있는 NS.NOBREAK.COM, NS2.NOBREAK.COM을 사용할수 있습니까?" "NS.NOBREAK.COM은 NOBREAK.COM 도메인 신청이 완료된 후 네임서버에서 설정 해주어야 사용할 수 있지 않습니까?"라는 의문이 들 수 있는데, 어떤 도메인을 하위 도메인으로 위임하기 위한 네임서버 정보는 상위 도메인에서 관리되기 때문에 가능하다. (참고: "글루 레코드")

2.4. 인버스 도메인(Inverse Domain) 신청

인버스 도메인은 IP에 대해 해당 도메인을 역으로 찾을 수 있도록 하는 서비스이다. 보통 ISP(Internet Service Provider)에서 IP를 할당받을 때 같이 신청한다. 다음과 같이 인버스 도메인에 대한 네임서버가 in-addr.arpa 네임스페이스에 등록되어 있는지 확인한다.

```
$ nslookup -type=ns 79.105.210.in-addr.arpa (C Class 210.105.79.x를 할당 받
았을 경우)
Server:  ns.nobreak.com
Address:  0.0.0.0

79.105.210.in-addr.arpa nameserver = ns.nobreak.com
79.105.210.in-addr.arpa nameserver = ns2.nobreak.com
ns.nobreak.com  internet address = 210.105.79.2
ns2.nobreak.com internet address = 210.105.79.3
```

만약 다음과 같은 메시지가 나온다면, 인버스 도메인 등록이 안되어 있는 것이므로, 해당 ISP에 신청하여야 한다.

```
*** ns.nobreak.com can't find 79.105.210.in-addr.arpa.: Non-
existent host/domain
```

2.5. Name Server 설정

다음은 nobreak.com 도메인에 대한 가상 네트워크 구성도 이다.

Figure 2-1. 네트워크 구성도

네트워크엔 서버가 3대 연결되어 있다. DNS를 구축하기 전에, 그림과 같이 미리 각 서버에 호스트명과 IP를 부여하자. 보통 네임서버는 ns(primary), ns2(secondary)를 호스트명으로 사용하고, IP 1(할프로 받았을 경우엔 129)을 라우터 혹은 스위치, 2를 NS, 3을 NS2에 할당한다. 도메인 NOBREAK.COM은 앞서 등록기관에 신청하였으니, NS.NOBREAK.COM, NS2.NOBREAK.COM에 네임서버 설정을 하면 된다.

2.5.1. BIND-4 부트 파일 named.boot

BIND-4 부트 파일 named.boot는 BIND 시동시 참조되며, 네임 데몬이 필요로 하는 환경정보와 운영할 도메인에 대한 Primary/Secondary 설정이 기술된다. 일반적으로 시스템의 `/etc/named.boot`에 위치하며, 대부분의 유닉스 스타트업 스크립트는 부트 파일이 존재하면 시동시 BIND를 자동으로 구동한다. 부트 파일상의 모든 엔트리는 반드시 1열 에서 시작하여야 하며 ';'은 주석을 의미한다.

```
directory  /var/named
```

directory 엔트리는 관련 파일들이 위치한 경로를 나타낸다. 이 경로는 부트 파일에 나타나는 파일들의 베이스 경로로 작용한다. 따라서 파일들은 본 경로를 기준으로 상대경로 표기해야 하며 여기서는 `/var/named` 디렉토리를 베이스 디렉토리로 한다. (대부분의 시스템 기본 베이스 경로는 `/var/named`, `/etc/named` 혹은 `/etc/namedb`이다)

```
cache  . named.root
```

cache 레코드는 말 그대로의 캐쉬가 아니라 루트 네임서버 정보가 들어있는 데이터베이스 파일을 나타낸다. BIND는 타 도메인 정보를 루트 네임서버에서부터 추적하는데, 이 루트 네임서버에 대한 정보를 본 파일에서 참조하게 된다. `/var/named/named.root`와 같이 베이스 디렉토리에 위치시키면 된다.

캐쉬 파일은 Internic(현재는 존재하지 않고 일부 서비스만이 남아있다)에서 배포하며 ftp://ftp.rs.internic.net/domain/named.root 에서 구할 수 있다. 루트 네임서버 목록이 자주 수정되진 않지만 조금씩 바뀌기 때문에, 한달 걸러 한번씩은 업데이트 해줄 필요가 있다. 스크립트를 Cron으로 정기적으로 수행해 캐쉬 파일을 업데이트하는 것도 좋은 방법이다.

```
primary  nobreak.com  zone-nobreak.com
```

해당 도메인에 대해 Primary 네임서버로 동작함을 말한다. 예는 nobreak.com 도메인에 대해 본 서버가 Primary 이며, 데이터베이스 파일은 `/var/named/zone-nobreak.com` 임을 나타낸다.

```
secondary  nobreak.com  210.105.79.2  sec-nobreak.com
```

도메인에 대해 Secondary 네임서버로 동작한다. 세 번째 필드는 Primary 네임서버의 IP 주소이고, 네 번째 필드는 Primary에서 전송 받은 zone 파일이 저장될 파일명이다. 210.105.79.2로부터 nobreak.com 도메인의 데이터베이스를 전송(Zone Transfer)받아 `/var/named/sec-nobreak.com`로 관리함을 뜻한다.

Figure 2-1의 네트워크에 대한 부트 파일은 다음과 같이 작성될 수 있다.

*** NS.NOBREAK.COM(Primary NS)의 /etc/named.boot** 파일

```
directory                           /var/named
cache      .                        named.root
primary    0.0.127.in-addr.arpa     zone-0.0.127.in-
addr.arpa    ; loopback
primary    79.105.210.in-addr.arpa  zone-79.105.210.in-
addr.arpa ; Reverse ZONE
primary    nobreak.com              zone-
nobreak.com            ; Forward ZONE
```

0.0.127.in-addr.arpa는 loopback 주소 127.0.0.1를 위한 것이다. loopback 주소가 사용되지 않는 시스템은 없기 때문에, 'primary 0.0.127.in-addr.arpa ...'와 같은 라인은 네임서버마다 갖고 있다. 그 다음 두 라인이 할당받은 C Class IP 블락 210.105.79와 도메인 nobreak.com 을 위한 설정이다.

*** NS2.NOBREAK.COM(Secondary NS)의 /etc/named.boot** 파일

```
directory                           /var/named
cache      .                        named.root
primary    0.0.127.in-addr.arpa     zone-0.0.127.in-addr.arpa
secondary  79.105.210.in-addr.arpa  210.105.79.2  sec-79.105.210.in-
addr.arpa
secondary  nobreak.com              210.105.79.2  sec-nobreak.com
```

Secondary 네임서버 설정이다. loopback은 Primary로 놓아둔다. Secondary의 설정은 이것이 전부이다. (`/var/named/zone-0.0.127.in-addr.arpa`는 있어야 함)

Secondary는 해당 도메인의 Primary에 접속하여 데이터 베이스를 전송받아, `sec-79.105.210.in-addr.arpa`, `sec-nobreak.com`으로 저장, 관리한다.

2.5.2. BIND-8 부트 파일 named.conf

BIND-4와 BIND-8의 관련 파일 작성법중 유일하게 차이가 나는 부분이 바로 이 부트 파일이다. BIND-8 부트 파일의 기본적인 구성은 BIND-4와 비슷하지만, 많은 부분 추가 확장되었기 때문에, 이를 수용하고 앞으로의 추가사항을 손쉽게 적용할 수 있도록 파일 포맷이 변경되었다. 그리고 구버젼 부트 파일과의 혼동을 막기위해 named.conf로 리네임 되었다. 어떻게 보면 C 언어의 문법과 매우 흡사한 것을 알 수 있다. 설정을 좀더 세밀하게 할 수 있도록 작성법이 바뀌었을 뿐, BIND-4의 부트 파일과 크게 다를 것은 없다. 다음은 앞서 작성한 BIND-4 기반 부트 파일을 BIND-8에 맞게 변환한 예이다. 일반적으로 BIND-8 기반의 부트 파일은 다음에 나열된 레코드정도만이 활용되지만, 재미난 부분이 많으므로 좀더 깊숙히 알고 싶다면 http://www.isc.org/products/BIND/docs/ 를 참고하기 바란다.

다음은 Primary 네임서버를 위한 부트 파일이다.

*** NS.NOBREAK.COM(Primary NS)의 /etc/named.conf** 파일

```
// 이것은 주석이다. BIND-8에서 ';'은 주석이 아니라, 라인의 끝을 의미한다.
options {
        directory "/var/named";             // Zone 파일의 베이스 디렉토리
        dump-file "/var/tmp/named_dump.db"; // Dump 파일이 생성되는 경로
        statistics-file "/var/tmp/named.stats"; // 통계 파일이 생성되는 경로
        pid-file  "/var/run/named.pid";     // 프로세스 ID가 담긴 파일 생성 경
로
};
logging {   // 불필요한 정보를 로그파일에 남기지 않는다.
        category lame-servers { null; };
        category cname { null; };
        category response-checks { null; };
        category notify { null; };
```

```
};

zone "." IN {                      // 캐쉬 파일
        type hint;
        file "named.root";
};
zone "0.0.127.in-addr.arpa" IN {    // localhost를 위한 Primary 도메인 설정
        type master;
        file "zone-0.0.127.in-addr.arpa";
};
zone "79.105.210.in-addr.arpa" IN { // 할당 IP 블락에 대한 Reverse Zone
        type master;
        file "zone-79.105.210.in-addr.arpa";
};
zone "nobreak.com" IN {             // 도메인 nobreak.com 에 대
한 Forward Zone
        type master;
        file "zone-nobreak.com";
};
```

Secondary 네임서버를 위한 부트 파일은 다음과 같이 작성된다.

*** NS2.NOBREAK.COM(Secondary NS)의 /etc/named.conf 파일**

```
options {
        directory "/var/named";
};
logging {
        category lame-servers { null; };
        category cname { null; };
};

zone "." IN {
        type hint;
        file "named.root";
};
zone "0.0.127.in-addr.arpa" IN {    // localhost를 위한 Primary 도메인 설정
        type master;
        file "zone-0.0.127.in-addr.arpa";
};
zone "79.105.210.in-addr.arpa" IN { // Reverse Zone에대한 Secondary 설정
        type slave;
        file "sec-79.105.210.in-addr.arpa";
        masters { 210.105.79.2; };  // Primary NS의 IP 주소
};
zone "nobreak.com" IN {             // nobreak.com 의 Secondary 설정
        type slave;
        file "sec-nobreak.com";
        masters { 210.105.79.2; };
};
```

2.5.3. 리소스 레코드(Resource Record)

Zone 파일은 Forward, Reverse 두 가지로 구분된다. Forward Zone은 도메인에 대한 IP 정보를 갖고 있는 데이터베이스이고, Reverse Zone은 IP에 대한 도메인정보를 갖는 데이터베이스이다. 앞서 `named.boot` 파일에 네임서버가 loopback, 79.105.210.in-addr.arpa, nobreak.com 도메인에 대해 Primary로 동작하도록 설정하였다. 이중 `zone-0.0.127.in-addr.arpa`와 `zone-79.105.210.in-addr.arpa`가 Reverse Zone 파일이고, `zone-nobreak.com`이 Forward Zone 파일이다. Zone 파일은 BIND-4와 BIND-8에서 작성법이 동일하다.

먼저 Figure 2-1의 네트워크 구성에 따라 Forward Zone 파일 `zone-nobreak.com`을 작성하여 보자.

2.5.3.1. SOA 레코드 (Start Of Authority)

Zone 파일은 항상 SOA 레코드로 시작한다. SOA 레코드는 해당 도메인, nobreak.com에 대해 네임서버가 인증(authoritative)된 자료를 갖고 있음을 의미하며, 자료가 최적의 상태로 유지, 관리될 수 있도록 한다.

```
nobreak.com. IN  SOA  ns.nobreak.com. hostmaster.nobreak.com. (
                      1998122800  ;Serial
                      21600       ;Refresh ( 6 hours)
                      1800        ;Retry   (30 minutes)
                      1209600     ;Expire  (14 days)
                      86400)      ;Minimum ( 1 day)
```

1열에는 해당 Zone 파일에 대한 도메인명이 들어간다. 도메인명 끝의 도트를 잊지 말자. 다음과 같이 도메인명 대신 '@' 표시를 사용하여도 된다.

```
@            IN  SOA  ns.nobreak.com. hostmaster.nobreak.com. (
```

IN(Internet)은 클래스명이다. HS, HESIOD, CHAOS와 같은 클래스도 존재하지만, 일반적으로 사용되지 않으므로 항상 IN이 사용된다고 생각하자.

SOA 다음엔 Primary 네임서버와 관리자 Email 주소가 들어간다. hostmaster.nobreak.com. 이 Email 주소인데, 일반적 Email 표기법에서 '@'를 도트로 바꾸어 쓰면 된다. 본 Email은 해당 도메인의 콘택 포인트(Responsible Person)로서 도메인에 문제가 발생할 경우 이를 리포팅하는 용도로 사용된다. Namespace를 좇으며 도메인 오류를 점검하는 lamers 와 같은 도구들은 문제가 검출되었을 때 본 Email로 통지하여 준다.

다음 괄호로 둘러싸인 부분엔 Serial, Refresh, Retry, Expire, Minimum 5개의 시간(초) 필드가 놓인다. Minimum을 제외한 4개 필드는 Secondary 네임서버를 제어하기 위한 값이다. 기본 단위는 '초'이고, 단위기호 M(Minute), H(Hour), D(Day), W(Week)를 붙여 30M, 8H, 2D, 1W와 같이 사용할 수도 있다.

- Serial: Serial은 Secondary가 Zone 파일의 수정여부를 알 수 있도록 하기 위함이다. Secondary는 백업본의 Serial이 Primary의 Serial보다 작을 경우 Zone 파일을 재전송 받는다. 따라서 Zone 파일이 수정된 후 Serial이 변경되지 않는다면, Secondary는 백업카피를 업데이

트하지 않음을 유의하자. Secondary가 없다면 Serial은 의미가 없지만 그렇다 할지라도 Zone 파일이 수정되었을 때 Serial을 증가하는 것은 좋은 습관이다.

- Serial의 표기는 증가하는 임의 숫자보단 일반적으로 최종 수정일을 YYYYMMDDNN의 형식으로 표기한다. YYYYMMDDNN 연도 표기법은 4294년까지 표기 가능하다.
- Refresh: Primary측의 Zone 데이터베이스 수정여부를 Secondary가 검사하는 주기이다. 네트워크의 변경이 잦아 Zone파일이 자주 수정된다면, 3H(10800) 정도로 설정한다. Zone이 안정되는 시점에서는 일반적으로 6H(21600) - 12H로 설정한다.
- Retry: Secondary측에서, Primary와 연결이 안될 경우, 재 시도 시간 주기이다. Refresh 기간보다 적을때 의미가 있으며, 대부분의 경우 30M(1800) - 1H로 설정한다.
- Expire: Secondary가 Expire로 지정된 시간동안 Primary에 연결하지 못할 경우, 오래된 백업카피의 자료가 더 이상 유효하지 않다고 보고, 해당 도메인에 대한 답변을 하지 않는다. 이 값을 너무 낮게 책정하는 것은 좋지 않다. 보통 1W - 2W(1209600)로 설정한다.
- Minimum: 타 네임서버가 본 Zone에 기술된 자료를 갖고 갔을 경우, 그 자료에 대한 유효기간(캐쉬에 살아있는 시간)을 설정한다. TTL(Time To Live)값이 명시되지 않은 레코드는 본 값을 기본으로 갖게 된다. 특정 레코드가 변경되었을 때, 이것이 인터넷에 전파되어 업데이트되는 주기는 전적으로 이 Minimum 값에 의존한다. 일반적으로 SOA에서는 1D(86400)를 설정하여 전체 레코드에 적용하고, 잦은 변경이 예상되는 레코드만 명시적으로 1H - 3H 정도로 낮추는 방법을 사용한다. 0은 캐싱을 하지 말라는 의미이다.

2.5.3.2. NS(Name Server) 레코드

NS 레코드로 해당 도메인에 대한 네임서버를 다음과 같이 나타낸다.

```
nobreak.com.    IN  NS      ns.nobreak.com.
                IN  NS      ns2.nobreak.com.
```

또 다른 NS의 활용으로는, 거대 도메인에서 서브 도메인을 다른 네임서버로 위임할 때이다. Namespace상의 가지연결은 이 NS 레코드로 이루어 지는데, 거대 도메인일 경우 해당하는 부분이므로, 여기서는 해당 도메인에 대한 위임 정보만을 나타낸다고 알아두자. 도메인 위임에서 자세히 다룬다.

2.5.3.3. A(Address) & CNAME(Canonical Name) 레코드

A 레코드는 도메인에 IP를 부여한다. 다음 설정을 보자. mail과 power에 A 레코드로 IP를 매핑 하였다. (mail과 mail.nobreak.com. 은 동일하게 해석된다.)

```
; Host addresses
mail.nobreak.com.   IN  A       210.105.79.2
power               IN  A       210.105.79.103
; Aliases
www                 IN  CNAME   power.nobreak.com.
ftp                 IN  CNAME   www
```

CNAME 레코드는 도메인에 대한 또 다른 이름이 가능하도록 한다. 예에서는 power.nobreak.com, www.nobreak.com, ftp.nobreak.com은 모두 같은 IP 210.105.79.103을 갖게 된다. ftp와 같이 CNAME이 CNAME을 포인팅 하는 경우는, 여러 DNS 관련 자료에서 다르게 얘기되고 있지만, 이것은 가능하다. CNAME은 포인팅하는 오리지널 도메인의 레코드를 모두 상속받기 때문에, CNAME으로 설정된 도메인은 추가 레코드를 갖을 수 없음을 유의한다. 또한, MX, NS 등의 레코드에도 CNAME으로 설정된 도메인을 넣어서는 안된다. 반드시 주의하여야 한다. CNAME의 잘못된 사용은 BIND 로그를 유심히 관찰하지 않으면 찾기 어려우므로, 확실히 할 수 없다면 CNAME으로 설정된 레코드를 아예 다른 레코드의 인자로 놓지 않는 것이 좋다. 숙련된 도메인 메니저 중에서도 트래픽과, 퍼포먼스라는 측면에서 CNAME을 전혀 사용하지 않는 경우도 있다. (참고: CNAME의 사용에 관해)

```
ftp                  IN   CNAME   www  ; (X) CNAME엔 추가레코드를 갖을 수
                     IN   MX      mail ;     없다.

power                IN   MX 10   mail ; (X) MX에 CNAME으로 설정된
mail                 IN   CNAME   ns   ;     레코드가 올 수 없다.
```

2.5.3.4. MX(Mail eXchanger) 레코드

MX 레코드는 해당 호스트의 메일 라우팅 경로를 조정한다. 다음과 같이 설정되어 있을 경우, account@nobreak.com 으로 보내어 지는 편지는 실제 mail.nobreak.com. 으로 전송된다. 만약 mail.nobreak.com. 에 연결할 수 없다면, 다음 우선순위인 power.nobreak.com 으로 편지를 배송하게 된다. MX Priority_Number 와 같이 사용하며, Priority_Number의 숫자는 적을수록 우선순위가 높다. MX 알고리즘에서 자세히 다룬다.

```
nobreak.com.         IN   MX   10   mail.nobreak.com.
                     IN   MX   20   power.nobreak.com.
mail.nobreak.com.    IN   A         210.105.79.2
power.nobreak.com.   IN   A         210.105.79.103
```

다음과 같이 MX 레코드에 CNAME으로 설정된 도메인을 넣으면 안된다. 이럴 경우 몇몇 MTA(Mail Transfer Agent: sendmail)는 메일 라우팅 경로를 찾지 못하여, 메일을 주고받을 수 없다. 이는 송신인이 사용하는 MTA의 종류와 버전에 의존적이므로, 경험 많은 도메인 메니저가 아니면 문제의 원인을 진단하기도 어렵다. 반드시 주의하자.

```
nobreak.com.         IN   MX   10   mail.nobreak.com.   ; (X) 잘못된 사용
mail.nobreak.com.    IN   CNAME     power.nobreak.com.
power.nobreak.com.   IN   A         210.105.79.103
```

2.5.3.5. PTR(Pointer) 레코드

PTR 레코드는 IP 주소에 대해 도메인명을 매핑하여 주며, Reverse Zone 파일에서 사용된다. 다음은 IP 210.105.79.2에 대한 설정 예이다.

```
2.79.105.210.in-addr.arpa.  IN   PTR   ns.nobreak.com.
```

Forward Zone에서는 다수의 도메인이 A(혹은 CNAME) 레코드를 통해 같은 IP를 갖을 수 있지만, PTR 레코드는 중복이 허용되지 않기 때문에, 해당 IP에 대한 대표 도메인명 하나만을 설정하여야 한다.

2.5.3.6. 기타 레코드들

Zone 데이터베이스에 필요한 레코드들은 위에 나열한 것만으로도 충분하지만, 더 많은 레코드들이 존재한다. 다음에 반드시 필요하지는 않으나, 종종 사용되는 레코드를 소개한다. 더 자세한 정보가 필요하다면 RFC1035 (ftp://ftp.openbird.com/pub/rfc/rfc1035.txt), RFC1183 (ftp://ftp.openbird.com/pub/rfc/rfc1183.txt), RFC2163 (ftp://ftp.openbird.com/pub/rfc/rfc2163.txt)을 참고하자.

```
power           IN  A      210.105.79.103
                IN  HINFO  "Sun Sparc Ultra 5"  "Solaris 2.6"
                IN  TXT    "Nobreak's Primary Server"
                IN  TXT    "WWW, FTP is now available"
                IN  RP     hostmaster.nobreak.com.  hostinfo.nobreak.com.
hostinfo        IN  TXT    "Seung-young Kim, +82-42-864-4440/1"
```

HINFO(Host INFOrmation) 레코드는 두 개의 문자열(CPU 정보, OS 정보)을 갖으며 시스템 정보를 나타낸다. 문자열에 공백이 포함되어 있을 경우에는 반드시 큰따옴표를 사용하여야 한다.

TXT(TeXT) 레코드는 텍스트 정보를 갖으며 중첩되어 사용될 수 있다. RP(Responsible Person)는 담당자의 정보를 표시하는데, Email 주소(@를 도트로 치환한)와, 담당자 정보(TXT 레코드를 갖는 도메인을 포인팅함)를 갖는다. HINFO를 포함한 몇몇 레코드는 보안을 이유로 사용치 말아야 한다는 의견도 있다.

2.5.4. Zone 데이터베이스 예제

Figure 2-1의 네트워크 구성에 대한 Forward Zone 파일 zone-nobreak.com은 다음과 같이 작성될 수 있다.

```
* nobreak.com 도메인에 대한 Forward Zone 파일 /var/named/zone-nobreak.com
@               IN      SOA     ns.nobreak.com. hostmaster.nobreak.com. (
                                1998122801  ;Serial
                                21600       ;Refresh ( 6 hours)
                                1800        ;Retry   (30 minutes)
                                1209600     ;Expire  (14 days)
                                86400)      ;Minimum ( 1 day)
                IN      NS      ns.nobreak.com.
                IN      NS      ns2.nobreak.com.
                IN      MX 10   mail            ; 메일 라우팅 호스트

mail            IN      A       210.105.79.2
```

```
; Hosts Here - This is comments
router          IN      A       210.105.79.1
ns              IN      A       210.105.79.2
ns2             IN      A       210.105.79.3
power           IN      A       210.105.79.103
                IN      HINFO   "Sun Sparc Ultra 5"         "Solaris 2.6"
                IN      TXT     "Nobreak Technologies, Inc."
www             IN      CNAME   power
```

인버스 도메인을 위한 Reverse Zone 파일 zone-79.105.210.in-addr.arpa은 다음과 같이 작성된다.

*** Reverse Zone 파일 /var/named/zone-79.105.210.in-addr.arpa**

```
@               IN      SOA     ns.nobreak.com. hostmaster.nobreak.com. (
                                1998122801  ;Serial
                                21600       ;Refresh ( 6 hours)
                                1800        ;Retry   (30 minutes)
                                1209600     ;Expire  (14 days)
                                86400)      ;Minimum ( 1 day)
                IN      NS      ns.nobreak.com.
                IN      NS      ns2.nobreak.com.
; IP-Domain mapping here
1               IN      PTR     router.nobreak.com.
2               IN      PTR     ns.nobreak.com.
3               IN      PTR     ns2.nobreak.com.
103             IN      PTR     power.nobreak.com.
```

loopback 주소를 위한 Reverse Zone 파일 또한 다음과 같이 작성된다. IP 127.0.0.1을 localhost.로 매핑하는 것이 전부이므로, 본 파일은 어느 네트워크에서나 비슷하게 작성될 것이다.

*** loopback을 위한 Reverse Zone 파일 /var/named/zone-0.0.127.in-addr.arpa**

```
@               IN      SOA     ns.nobreak.com. hostmaster.nobreak.com. (
                                1998122801  ;Serial
                                21600       ;Refresh ( 6 hours)
                                1800        ;Retry   (30 minutes)
                                1209600     ;Expire  (14 days)
                                86400)      ;Minimum ( 1 day)
                IN      NS      ns.nobreak.com.
                IN      NS      ns2.nobreak.com.
; IP-Domain mapping here
1               IN      PTR     localhost.
```

Zone 파일에서의 도메인 표기는 반드시 FQDN 표기법을 따라야 한다. BIND는 도트로 끝나지 않는 문자열은 호스트명으로 처리하므로, ns.nobreak.com 을 ns.nobreak.com.nobreak.com.으로 해석한다. 따라서 ns.nobreak.com. 과 같이 도트를 붙인 FQDN으로 표기하거나, ns 와 같이 호스트명만 사용하여야 한다. 도트를 빼먹는 실수는 매우 빈번히 발생하므로, 주의하자.

2.5.4.1. 호스팅 업체를 위한 Zone 데이터베이스 예제

호스팅업체의 경우 다음과 같이 다수의 도메인을 하나의 Zone 데이터베이스로 관리할 수가 있다. 만약, 호스팅 도메인별로 별도의 Zone을 유지한다면, 호스팅 서버의 IP 변화와 같이 관련된 모든 Zone이 수정되어야 하는 상황이 오지 않기를 기도하거나, 반나절을 편집기와 씨름할수 있는 끈기를 배워야할 것이다. 여기서 소개하는 팁은 사용자가 많은 호스팅 업체일수록 유용하게 활용될 수 있으며, 도메인 추가/수정/삭제에 드는 시간과 노력을 절약할 수 있을 것이다.

*** named.boot (BIND-4)**

```
primary         netbsd.org                      zone-default
primary         openbsd.org                     zone-default
...
primary         freebsd.org                     zone-freebsd.org ; 별도의 추
가 도메인이 필요한 경우
...
```

*** named.conf (BIND-8)**

```
zone "netbsd.org"       IN { type master; file "zone-default"; };
zone "openbsd.org"      IN { type master; file "zone-default"; };
...
zone "freebsd.org"      IN { type master; file "zone-freebsd.org"; };
...
```

*** zone-default**

```
@               IN      SOA     ns.nobreak.com. hostmaster.nobreak.com. (
                                1999030601  ;serial
                                21600       ;Refresh ( 6 hours)
                                1800        ;Retry   (30 minutes)
                                1209600     ;Expire  (14 days)
                                86400)      ;Minimum ( 1 day)
                IN      NS      ns.nobreak.com.
                IN      NS      ns2.nobreak.com.
                IN      A       210.105.79.39
                IN      MX 10   @
                IN      MX 20   mqueue.nobreak.com.

www             IN      CNAME   @
telnet          IN      CNAME   @
ftp             IN      CNAME   @
mail            IN      CNAME   @
pop             IN      CNAME   @
news            IN      CNAME   news.nobreak.com.
```

*** zone-freebsd.org**

```
$INCLUDE zone-default

ftp.kr          IN      A       147.46.102.39
www.kr          IN      CNAME   @
```

2.6. Name Server 구동

네임서버를 구동하기 위한 설정이 마무리되었다. 관련 파일들이 준비되었음을 확인한 후, 네임서버를 구동하자. 일련의 과정은 다음과 같다.

```
* NS.NOBREAK.COM (BIND-8)
# ls /etc/named.conf     # BIND-8 부트 파일
/etc/named.conf
# ls /var/named/         # Zone 데이터베이스 파일 확인
named.root
zone-0.0.127.in-addr.arpa
zone-79.105.210.in-addr.arpa
zone-nobreak.com
# /usr/sbin/named        # 네임 데몬 구동 (Solaris: /usr/sbin/in.named)
# ps ax | grep named     # 프로세스 동작 확
인 (Solaris: ps -e | grep in.named)
  254  ?  S    0:00 named

* NS2.NOBREAK.COM (BIND-4)
# ls /etc/named.boot      # BIND-4 부트 파일
/etc/named.boot
# ls /var/named/          # Zone 데이터베이스 파일 확인
named.root
zone-0.0.127.in-addr.arpa
# ndc start               # ndc(Name Daemon Control)가 설치되어 있을 경우
Name Server Started
# ndc status
  254  ?  S    0:00 named
# ls /var/named/          # Primary의 Zone 전송여부 확인
named.root
sec-79.105.210.in-addr.arpa
sec-nobreak.com
zone-0.0.127.in-addr.arpa
```

2.7. Name Server 동작 확인

여기에선 [그림 3]의 가상 네트워크 구성도에 따른 설정을 다루었지만, 기본적으로 필요한 부분은 모두 적용되어 있으므로 실제 네트워크에 적용할 때에도 같은 느낌으로 설정하면 된다. 다음과 같이 타 네임서버를 통해 질의를 던져봄으로써, Namespace 가지상에 잘 연결되어 있음을 확인하자.

```
$ nslookup power.nobreak.com ns.nobreak.com   # Primary 동작 확인
Server:  ns.nobreak.com
Address:  0.0.0.0

Name:    power.nobreak.com
```

```
Address:  210.105.79.103

$ nslookup power.nobreak.com ns2.nobreak.com  # Secondary 동작 확인
Server:  ns2.nobreak.com
Address:  210.105.79.3

Name:    power.nobreak.com
Address:  210.105.79.103

$ nslookup power.nobreak.com ns.kornet.ne.kr  # Namespace 링크 확인
Server:  ns.kornet.nm.kr
Address:  168.126.63.1

Name:    power.nobreak.com
Address:  210.105.79.103
```

위의 3가지 질의가 성공적으로 수행되었다면, 일단 네임서버가 정상적으로 운용된다고 생각해도 좋다. 확실히 하기 위해선, 로그파일 분석을 통해 Zone 데이터베이스 구성상의 오류와 BIND의 동작 상태를 살펴보아야 한다.

3Chapter. 클라이언트(Resolver) 설정

3.1. Unix 시스템에서의 클라이언트 설정

Unix 시스템상에서 DNS를 사용하기 위해서는 `/etc/resolv.conf` 파일과, `/etc/nsswitch.conf` 파일을 손보아야 한다. `/etc/resolv.conf`에는 도메인 Resolving시 쿼리를 던질 네임서버를 3개까지 나열할 수 있다. 일반적으로 로컬 Primary, Secondary, ISP 네임서버 3개를 다음과 같이 명시한다.

```
search nobreak.com       # 도메인 서픽스(Suffix)
nameserver 210.105.79.2  # ns.nobreak.com
nameserver 210.105.79.3  # ns2.nobreak.com
nameserver 168.126.63.1  # ns.kornet.ne.kr (사용하는 ISP의 NS)
```

’search nobreak.com’은 호스트명만 주었을 경우 서픽스로 사용할 도메인을 뜻한다. 다른 도메인의 호스트에 자주 접속하거나, 여러 도메인을 관리한다면, 해당 도메인을 다음과 같이 여럿 놓을 수도 있다.

```
search nobreak.com
search kr.freebsd.org
```

이런 경우엔 호스트명 www는 먼저 www.nobreak.com으로 확장되어 검색되고, 검색되지 않을 경우 www.kr.freebsd.org로 확장되어 다시 검색된다. 잘못된 호스트명을 주었을때도, 가능한 조합을 모두 검색하므로, 많은 도메인을 search 리스트에 포함하는 것은 바람직하지 않다.

자주 사용하지는 않지만 도메인 별로 네임서버를 다르게 설정할 수도 있다. 주로 네임서버의 캐쉬 영향을 받지 않고, 해당 도메인의 수정사항을 즉각 적용하기 위해 쓰인다.

```
search nobreak.com

domain nobreak.com
nameserver 210.105.79.2
...

domain kornet.ne.kr
nameserver 168.126.63.1
...
```

호스트 네임을 DNS로 확장하여 검색하기 위해서는 `/etc/nsswitch.conf` 파일의 라인 ’hosts:’에 ’dns’가 추가되어 있어야 한다. 근래 대부분의 OS는 기본적으로 설정되어 배포되지만, 확실히 하기 위해 확인하자.

```
hosts:      files dns
```

files 는 호스트명을 `/etc/hosts` 파일에서 검색함을 뜻한다. 시스템은 주어진 순서(files, dns)로 검색을 시도한다.

3.2. Microsoft Windows 시스템에서의 클라이언트 설정

Windows에서 Unix 시스템의 `/etc/resolv.conf` 파일에 해당하는 설정은 '네트워크'에서 할 수 있다. '제어판'의 '네트워크' 메뉴중 TCP/IP 프로토콜의 등록정보를 열면, Figure 3-1와 같은 DNS 설정탭을 볼 수 있다.

Figure 3-1. Windows NT에서의 **Resolver** 설정 예

DNS에 등록된 PC의 호스트명을 '호스트 이름'에 넣는다. Unix 시스템의 `resolv.conf` 파일과 비교하여, '도메인'은 'domain'에, 'DNS 서비스 찾기 순서'는 'nameserver'에, '도메인 서픽스 찾기 순서'는 'search' 리스트에 해당한다.

4Chapter. 도메인 위임 (Domain Parenting & Delegation)

여러 계열사를 갖는 대기업의 경우, 계열사를 www.core.nobreak.com 과 같이 서브 도메인으로 분류하여 운영할 필요가 있다. 또한 해당 계열사의 네트워크가 방대할 경우, 효율적인 관리를 위해 해당 도메인을 해당 계열사로 위임할 수도 있다.

Figure 4-1. Domain Delegation에 의한 **Namespace** 계층도

Figure 4-1를 보자.FreeBSD.ORG 도메인은 각 국가별로 서브 도메인을 위임하여 운영하고 있고, KR.FreeBSD.ORG 도메인은 각 프로젝트 팀을 CORE 도메인아래 위치시킴으로 써, doc-kr.core.kr.FreeBSD.org 와 같은 계층구조를 유지한다. 이 모든 가지를 하나의 네임서버에서 관리 할 수도 있지만 네트워크의 규모가 불어남에 따라 행해지는 도메인 위임 관리는 전체 네트워크를 부드럽게 한다.

4.1. 부모 Zone에서의 서브 도메인 생성

서브 도메인이 반드시 위임되어야 하는 것은 아니다. 다음과 같이 부모 Zone 데이터베이스에 함께 정의될 수도 있다. 여기서는 회사내 망관리 그룹을 nms.nobreak.com 으로 분류하고, 그 아래 DNS 도메인을 생성한 경우이다.

```
* zone-nobreak.com (NS.NOBREAK.COM)
nms                         IN      MX 10   mail.nms
                            IN      MX 20   mail2.nms
dns.nms                     IN      A       150.183.110.53
                            IN      MX 10   mail.nms
                            IN      MX 20   mail2.nms
mail.nms                    IN      A       150.183.110.10
mail2.nms                   IN      A       150.183.110.11
```

nobreak.com 의 SOA 레코드가 그대로 적용되기 때문에, 생성된 서브 도메인 nms.nobreak.com 에 대한 SOA 레코드는 필요가 없다.

4.2. 서브 도메인 위임

nms.nobreak.com 의 규모가 커져 관리를 분산시키기 위해 타 네임 서버로 해당 도메인을 위임하고자 할 때에는 NS 레코드를 사용해 해당 도메인을 레퍼런싱 해주어야 한다. 다음은 nobreak.com 도메인에서 nms 를 ns.nms.nobreak.com 으로 위임한 예이다.

*** zone-nobreak.com (NS.NOBREAK.COM)**

```
nms                     IN      NS      ns.nms        ; Delegation
                        IN      NS      ns2.nms
ns.nms                  IN      A       150.183.110.2 ; Glue Record
ns2.nms                 IN      A       150.183.110.3
```

IP 주소는 A 레코드에서만 설정 할 수 있으므로, 반드시 글루 레코드를 사용하여 NS에 연결하여야 한다. (참고: 글루 레코드)

네임서버 ns.nms.nobreak.com 은 부트파일에 다음과 같이 위임받은 도메인에 대한 primary 설정이 필요하다.

*** named.boot (NS.NMS.NOBREAK.COM)**

```
primary         nms.nobreak.com         zone-nms.nobreak.com
```

그리고, Zone 데이터베이스 `zone-nms.nobreak.com`을 다음과 같이 작성한다.

*** zone-nms.nobreak.com (NS.NMS.NOBREAK.COM)**

```
@               IN      SOA     ns.nms.nobreak.com. hostmas-
ter.nms.nobreak.com. (
                        1999012901  ;Serial
                        21600       ;Refresh ( 6 hours)
                        1800        ;Retry   (30 minutes)
                        1209600     ;Expire  (14 days)
                        86400)      ;Minimum ( 1 day)
                IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
                IN      MX 10   mail
                IN      MX 20   mail2

ns              IN      A       150.183.110.2
ns2             IN      A       150.183.110.3
mail            IN      A       150.183.110.10
mail2           IN      A       150.183.110.11

dns             IN      A       150.183.110.53
                IN      MX 10   mail
                IN      MX 20   mail2
```

네임서버 ns2.nms.nobreak.com 를 Secondary로 설정하기 위해 부트파일에 다음을 추가한다.

*** named.boot (NS2.NMS.NOBREAK.COM)**

```
secondary       nms.nobreak.com  150.183.110.2  sec-nms.nobreak.com
```

이제 ns, ns.nms, ns2.nms의 BIND를 모두 행업한후, 외부 네임서버를 통해 동작을 확인한다.

```
$ nslookup  -type=NS  nms.nobreak.com  ns.kornet.ne.kr
nms.nobreak.com nameserver = ns.nms.nobreak.com
nms.nobreak.com nameserver = ns2.nms.nobreak.com

$ nslookup  dns.nms.nobreak.com  ns.kornet.ne.kr
Name:    dns.nms.nobreak.com
Address:  150.183.110.53
```

4.3. 옥텟 기반 in-addr.arpa 위임

nobreak.com 이 B 클래스 150.183을 다음과 같이 할당받았고, nms.nobreak.com 은 그중 서브넷 150.183.110을 사용한다고 가정하자.

```
$ nslookup  -type=NS  183.150.in-addr.arpa
183.150.in-addr.arpa    nameserver = ns.nobreak.com
183.150.in-addr.arpa    nameserver = ns2.nobreak.com
```

옥텟 단위의 Inverse 도메인 위임은 서브 도메인 위임과 같은 느낌으로 수월하게 이루어진다. 다음은 183.150.in-addr.arpa 에서 110.183.150.in-addr.arpa 을 위임하기 위한 Zone 설정이다.

```
* zone-183.150.in-addr.arpa (NS.NOBREAK.COM)
110             IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
```

그리고, ns.nms.nobreak.com 과 ns2.nms.nobreak.com 에 primary, secondary 설정을 넣고 Inverse Zone 파일을 생성함으로써, 인버스 도메인 위임이 마무리 된다.

```
* named.boot (NS.NMS.NOBREAK.COM)
primary         110.183.150.in-addr.arpa        zone-110.183.150.in-
addr.arpa

* named.boot (NS2.NMS.NOBREAK.COM)
secondary       110.183.150.in-addr.arpa        150.183.110.2   sec-
110.183.150.in-addr.arpa

* zone-110.183.150.in-addr.arpa (NS.NMS.NOBREAK.COM)
@               IN      SOA     ns.nms.nobreak.com. hostmas-
ter.nms.nobreak.com. (
                        1999012901  ;Serial
                        21600       ;Refresh ( 6 hours)
                        1800        ;Retry   (30 minutes)
                        1209600     ;Expire  (14 days)
                        86400)      ;Minimum ( 1 day)
                IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
```

```
2               IN      PTR     ns.nms.nobreak.com.
3               IN      PTR     ns2.nms.nobreak.com.
10              IN      PTR     mail.nms.nobreak.com.
11              IN      PTR     mail2.nms.nobreak.com.
53              IN      PTR     dns.nms.nobreak.com.
```

4.4. Classless in-addr.arpa 위임

참고: RFC2317 (ftp://ftp.openbird.com/pub/rfc/rfc2317.txt)

도메인 위임은 도트로 분리되는 세그먼트 혹은 옥텟(octet) 기반으로 설계되었기 때문에, 옥텟 기반의 인버스 도메인 위임엔 별다른 문제가 없다. 하지만 세그먼트를 나누어 210.105.79.1-128 까지는 nobreak.com 이 129-256까지는 nms.nobreak.com 이 사용하는 경우에는 인버스 도메인 위임이 수월치 않다.

같은 기관이 편의상 나누는 것이라면, 인버스 도메인에 한해 위임치 않고, 중앙에서 통합관리하는 것이 가능하나, IP 고갈로 인해 ISP들이 클래스를 할프(half) 또는 더 잘게 나누어 배급하고 있으므로, 이러한 기관들 사이에서는 한쪽 기관의 피해가 없도록 위임이 배려되어야 하겠다.

첫번째 방법으로는 79.105.210.in-addr.arpa의 설정에 있어서 129-256(B에 할당된)까지의 IP 블럭을 NS 레코드를 사용해 모두 네임서버 ns.nms.nobreak.com 로 레퍼런싱 해준다. 79.105.210.in-addr.arpa 에 대한 Zone 데이터 베이스 작성 예를 보자.

```
* zone-79.105.210.in-addr.arpa (NS.NOBREAK.COM)
2               IN      PTR     ns.nobreak.com.
3               IN      PTR     ns2.nobreak.com.
...
129             IN      NS      ns.nms.nobreak.com.   # 129-254까지 위임
                IN      NS      ns2.nms.nobreak.com.
...
254             IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
```

NS 레코드를 사용해 페어린팅 했기 때문에, ns.nms.nobreak.com 의 부트 파일에는 IP 129-254에 대해 다음과 같이 모두 primary 설정이 들어가 있어야 한다. 조금 번거로운 면이 있다.

```
* named.boot (NS.NMS.NOBREAK.COM)
primary  129.79.105.210.in-addr.arpa  zone-129.79.105.210.in-addr.arpa
...
primary  254.79.105.210.in-addr.arpa  zone-254.79.105.210.in-addr.arpa
```

또한, 각 Zone 파일 129.79.105.210.in-addr.arpa - 254.79.105.210.in-addr.arpa는 모두 다음과 같은 형식으로 작성되어야 한다.

```
* zone-129.79.105.210.in-addr.arpa (NS.NMS.NOBREAK.COM)
@               IN      SOA     ns.nms.nobreak.com. hostmas-
ter.nms.nobreak.com. (
                        1999012901  ;Serial
                        21600       ;Refresh ( 6 hours)
                        1800        ;Retry   (30 minutes)
                        1209600     ;Expire  (14 days)
                        86400)      ;Minimum ( 1 day)
                IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
                IN      PTR     dns.nms.nobreak.com.
```

```
* zone-130.79.105.210.in-addr.arpa (NS.NMS.NOBREAK.COM)
@               IN      SOA     ns.nms.nobreak.com. hostmas-
ter.nms.nobreak.com. (
                        1999012901  ;Serial
                        21600       ;Refresh ( 6 hours)
                        1800        ;Retry   (30 minutes)
                        1209600     ;Expire  (14 days)
                        86400)      ;Minimum ( 1 day)
                IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
                IN      PTR     dhcp.nms.nobreak.com.
```

IPv4는 4개의 옥텟으로 구분되고, 이것은 마지막 4번째 옥텟에 대한 Zone 이므로, 추가적인 정의가 있을 수 없다.

이 방법은 위임된 IP 개수만큼의 Zone 데이터베이스가 요구되기 때문에, 번거로워 실제로는 거의 사용하지 않는데, 도메인 페이런팅의 한계와 느낌을 전달하고자 소개하였다.

여기 조금 세련된 방법이 있다.

```
* zone-79.105.210.in-addr.arpa (NS.NOBREAK.COM)
2               IN      PTR     ns.nobreak.com.
3               IN      PTR     ns2.nobreak.com.
...
129             IN      CNAME   129.nms.79.105.210.in-addr.arpa.
...
254             IN      CNAME   254.nms.79.105.210.in-addr.arpa.
nms             IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.
```

Namespace 상에서 Inverse 도메인을 위한 in-addr.arpa 가지와, 일반적인 문자 도메인 가지의 기술적 차이가 전혀 없음을 상기하자. 편의상 arpa 도메인을 IP 매핑용으로 사용하는 것 뿐이니, nms.79.105.210.in-addr.arpa 와 같은 도메인을 만들고 패어런팅하는 것이 가능하고, 여기서는 이점을 이용해 Classless in-addr.arpa를 해결하고 있다.

즉 129.79.105.210.in-addr.arpa 는 129.nms.79.105.210.in-addr.arpa 를 CNAME 하는데, nms.79.104.210.in-addr.arpa 의 primary는 ns.nms.nobreak.com 이므로, 결국 위임은 성공적으로 성립된다.

*** named.boot (NS.NMS.NOBREAK.COM)**

```
primary  nms.79.105.210.in-addr.arpa  zone-nms.79.105.210.in-addr.arpa
```

*** zone-nms.79.105.210.in-addr.arpa (NS.NMS.NOBREAK.COM)**

```
@               IN      SOA     ns.nms.nobreak.com. hostmas-
ter.nms.nobreak.com. (
                                1999012901  ;Serial
                                21600       ;Refresh ( 6 hours)
                                1800        ;Retry   (30 minutes)
                                1209600     ;Expire  (14 days)
                                86400)      ;Minimum ( 1 day)
                IN      NS      ns.nms.nobreak.com.
                IN      NS      ns2.nms.nobreak.com.

129             IN      PTR     dns.nms.nobreak.com.
130             IN      PTR     mail.nms.nobreak.com.
...
254             IN      PTR     layla.nms.nobreak.com.
```

129-254까지의 PTR을 여기에 모두 나열한다. 79.105.210.in-addr.arpa Zone에서 각 IP를 모두 CNAME으로 잡아주어야 하는 불편함을 빼면 현재의 DNS 알고리즘상에선 최선의 방안이 아닌가 한다.

서로 다른 기관으로 할당된 쪼개어진 하나의 Class에 대한 Reverse 도메인 위임은 ISP 단계에서 이루어져야 하지만, 실제 ISP들은 Class 단위로만 위임처리를 하는경우가 대부분이여서, 해당 Class 전체를 한쪽 기관으로 위임하는 경우가 대부분이다. 따라서 이러한 경우에는 위임받은 기관에서 타 기관으로 할당된 IP 블락을 소개한 방법과 같이 레퍼런싱 해주어 타 기관의 피해가 없도록 하자. 네임서버를 비롯하여 Public IP를 갖는 호스트(PC를 포함하여)가 Reverse 매핑이 안되어 있을 경우에는 몇몇 서비스의 사용이 거부될 수도 있으며, 해당 호스트로의 Reverse Resolving 딜레이가 증가하는등 크고 작은 잠재적 문제가 야기된다.

Classless in-addr.arpa 위임과는 조금 다르게, 다수의 C 클래스 210.105.79 - 210.105.81 를 할당받았고, 이중 81 네트워크를 위임코자할 경우를 생각해보자. 한가지 방법은, 해당 ISP에 81.105.210.in-addr.arpa에 대한 네임서버 변경 신청만을 하는 것이겠지만, 보통은 이렇게 하지 않고, 여기서 설명하는 것과 같이 Namespace상의 링크가 해당 네트워크의 네임 서버에의해 포인팅 되도록하여, 네트워크의 변동에 빠르게 대처토록 한다.

5Chapter. 고급 기능

5.1. DNS Notify

참고: RFC1996 (ftp://ftp.openbird.com/pub/rfc/rfc1996.txt)

BIND-8 부터 지원하는 DNS Notify기능은 Primary의 Zone 데이터베이스가 수정되었음을 해당 Zone의 Authority를 갖는 Secondary 서버로 알려주어 Primary와 Secondary 네임서버의 동적 동기화를 가능케 한다.

BIND-4나 BIND-8에서 본 옵션을 사용하지 않으면 Zone의 SOA 영역에 명시된 Refresh를 주기로 Secondary가 Primary의 Serial 증가를 체크하여 Zone Transfer 하는 형태이나. BIND-8의 DNS Notify를 사용하면 하여 도메인 수정 변경에 따른 전파시간 (Maximum Propagation Delay)을 최소화 해준다.

Primary의 Zone이 업데이트 되면 BIND는 해당 Zone의 NS 레코드를 분석하여 자신을 제외한 나머지 네임서버에 Zone이 업데이트 되었음을 알리는 Notify 신호를 보내고, Secondary는 Primary Zone의 Serial이 증가하였음을 확인한후 Zone transfer를 통해 해당 Zone을 업데이트한다. 만약 Secondary가 DNS Notify를 지원하지 못한다면 "Not Implemented" 응답과 함께 해당 요청을 무시한다.

Figure 5-1. Maximum Propagation Delay

[큰 그림 보기 (images/b-dns-notify.gif)]

BIND-8는 기본적으로 DNS Notify가 켜져있다. 따라서 다음과 같은 설정을을 통해 전체 혹은 특정 Zone에 대해서 DNS-Notify를 적용치 않을 수 있다.

```
options {
notify no;  // 전체에 대해서 기능을 끈다. (디폴트 yes)
}

zone "freebsd.org" {
type master;
file "zone-freebsd.org"
notify no;          // 해당 도메인에 대해서만 기능을 끈다.
};

zone "freebsd.org" {
type master;
```

```
file "zone-freebsd.org"
notify yes;          // 해당 도메인에 대해서만 기능을 켠다.
};
```

Authority를 갖지 않는 네임서버에 Notify 리스트에 포함하고자 할 경우엔 also-notify 옵션을 사용한다.

```
zone "freebsd.org" {
type master;
file "zone-freebsd.org"
notify yes;
also-notify {210.124.149.130;};
};
```

RFC1996 (ftp://ftp.openbird.com/pub/rfc/rfc1996.txt)에 따르면 DNS Notify 요청을 받은 Secondary 네임서버는 해당 도메인의 Authority를 갖는 다른 네임서버에게 다시 DNS Notify 신호를 보내야 하는데 실제 BIND-8 구현에서는 포함되지 않았음을 참고하기 바란다. 이 기능은 네트워크 토폴로지상 Secondary가 Primary에 바로 접속치 못하고 다른 Secondary를 마스터로 설정하는 경우를 대비해 규정되었으나, 실용적으로 이러한 경우가 매우 드물고 바람직하지 않은 구성(Maximum Propagation Delay 증가)이기에 BIND-8에 같이 구현되지 않은듯 싶다.

5.2. Dynamic Update

참고: RFC2136 (ftp://ftp.openbird.com/pub/rfc/rfc2136.txt)

BIND-8 부터 지원되는 Dynamic Update는 해당 도메인의 Authority를 갖는 네임서버를 통해 Zone 파일을 수정치 않고도 레코드를 동적으로 원격 갱신할 수 있도록 한다. 도메인 관리를 자동화 하거나, 사용자별로 접속 도메인을 실시간 변경하여 제공하거나, DHCP에서의 주소-IP 매칭등과 같이 실시간 적으로 레코드가 변경, 갱신 될 필요가 있는 서비스에 특히 유용할 수 있겠다.

Dynamic Update는 보안을 이유로 기본적으로 기능이 꺼져있기 때문에 허용할 도메인에 대해 allow-update 옵션을 추가해야 한다.

```
zone "freebsd.org" {
type master;
file "zone-freebsd.org";
allow-update { 210.124.149.130; };
}
```

Dynamic Update는 BIND 배포판에 포함되어 있는 `nsupdate` 도구를 사용하여 명령행(non-interactive) 혹은 대화형(interactive)으로 조작이 가능하다. 대화형 모드에서 주어진 명령문은 묶음(조건문과 명령문)으로 실행이 가능하기 때문에 입력한 명령문(들)은 공백 라인에서 엔터를 한번 더 입력하여야 한다. 명령행 모드는 명령문을 주어진 파일이나 stdin 에서 입력받는다. 다음은 `nsupdate`에서 사용가능한 명령문이다.

prereq yxdomain DOMAIN-NAME

> DOMAIN-NAME이 존재(하나이상의 레코드가 설정되어 있음)함을 연속된 명령의 선행 조건으로 삼는다.

prereq nxdomain DOMAIN-NAME

> DOMAIN-NAME에 어떠한 레코드도 설정되어 있지 않음을 연속된 명령의 선행 조건으로 삼는다.

prereq yxrrset DOMAIN-NAME [CLASS] TYPE [DATA]

> DOMAIN-NAME에 해당 레코드가 존재함을 연속된 명령의 선행 조건으로 삼는다. DATA가 명시되어 있을 경우에는 정확하게 매칭이 되는 경우에만 조건이 성립된다.

prereq nxrrset DOMAIN-NAME [CLASS] TYPE

> DOMAIN-NAME에 해당 레코드가 존재하지 않음을 연속된 명령의 선행 조건으로 삼는다.

update delete DOMAIN-NAME [CLASS] [TYPE [DATA...]]

> TYPE이 명시되지 않았을 경우엔 해당 DOMAIN-NAME에 소속된 레코드를 모두 삭제한다. TYPE이 명시될 경우엔 매칭되는 레코드만이 제거된다.

update add DOMAIN-NAME TTL [CLASS] TYPE DATA...

> 지정된 레코드를 해당 도메인에 추가한다.

```
$ nsupdate
> update add freefall.freebsd.org. 3600 IN A 210.124.149.150
> [Enter]
...(messages)...
> ^D

$ nsupdate
> prereq nxrrset freebsd.org. IN MX
> update add freebsd.org. 3600 IN MX 10 mail.freebsd.org.
> [Enter]
...(messages)...
> ^D
```

기존에 A, CNAME 등의 레코드가 설정된 도메인명에 대해서 delete를 수행치 않고 add 명령을 입력했을때 기존 레코드의 데이터가 입력된 레코드의 데이터로 교체될거라는 생각은 하지 말아야 한다. 중복된 A 레코드의 입력은 도메인에 여러개의 IP를 매핑할 것이고, CNAME이 설정된 도메인명은 다른 레코드가 존재할 수 없음에도 CNAME을 add하는 명령이 해당 도메인의 A, MX와 같은 레코드를 자동으로 제거해주지는 않기 때문이다.

Dynamic Update를 통해 수정된 내역은 즉시 적용되며 `named`가 종료될 때 해당 Zone 데이터베이스에 직접 기록되어 다음번 구동시에도 그 내역이 변함없이 적용될 수 있도록 한다.

Dynamic Update에 대한 요청이 해당 도메인의 Authority를 갖는 Secondary로 보내어 졌다면 Secondary 네임서버는 Primary 네임서버로 요청을 전달하도록 되어있다. 물론 이러할 경우엔 Primary의 allow-update 억세스 리스트에는 Secondary가 포함되어 있어야 한다.

5.3. 미러링 서버간의 부하 공유

서버가 히트수를 감당하지 못할 경우, 그 해결책으로써 다수의 미러링 서버를 운영하여, 부하를 분담시키는 방법을 생각할 수 있다. 하지만, 이러한 방법은 미러링 서버를 사용자에게 홍보하여 서버의 부하가 이동하는데 실질적으로 많은 시간이 소요되고, 적절한 부하 분배를 기대하기가 힘이든 문제가 있다. 그래서 전화국의 대표 번호 서비스와 같이, 사용자의 요청을 각각의 미러링 서버로 연결해주는 대표 도메인을 생각할 수 있는데, 여기에서 그 방법을 소개한다. Shuffle Addresses이라 불리는 이 특별한 기법은 BIND 4.9 부터 지원된다.

```
www             180     IN      A       210.105.79.101
                180     IN      A       210.105.79.102
                180     IN      A       210.105.79.103
```

하나의 호스트명에 여러개의 IP주소를 주었을 경우, 네임서버는 해당 도메인에 대해 다음과 같이 라운드 로빈 방식으로 응답 한다.

```
$ nslookup www.nobreak.com
Name:    www.nobreak.com
Addresses:  210.105.79.101, 210.105.79.102, 210.105.79.103

$ nslookup www.nobreak.com
Name:    www.nobreak.com
Addresses:  210.105.79.102, 210.105.79.103, 210.105.79.101

$ nslookup www.nobreak.com
Name:    www.nobreak.com
Addresses:  210.105.79.103, 210.105.79.101, 210.105.79.102
```

이것이 로드 발랜싱(Load Balancing)은 아니지만, 클라이언트는 3대의 서버에 어느정도 공평하게 접속되므로, 부하를 공유하는 효과를 얻을 수 있고, 또한 외부로는 대표 도메인만을 알리면 되므로, 서버의 확장 및 축소에 유연하다. 본 기법을 적용할 때에는 라운드 로빈이 지원되지 않는 네임서버를 고려하여 TTL을 낮게 책정(TTL이 만기하여 다시 요청이 들어오도록) 하는 것도 좋다.

또하나의 방법으로는 다수의 CNAME을 연결하는 방법이다. 원칙적으로 다수의 CNAME은 거부되기 때문에, 반드시 다음과 같이 부트 파일에 별도의 옵션을 주어야 한다.

```
options {
        multiple-cnames yes;
};
```

이것은 BIND-8 에서만 가능하며, 대표적으로 YAHOO!(www.yahoo.com)가 이러한 방법으로 운영된다.

```
www             180     IN      CNAME   www1.nobreak.com.
                180     IN      CNAME   www2.nobreak.com.
                180     IN      CNAME   www3.nobreak.com.
www1            180     IN      A       210.105.79.101
www2            180     IN      A       210.105.79.102
www3            180     IN      A       210.105.79.103
180 IN A 210.105.79.104
180 IN A 210.105.79.105
```

다수의 A 레코드 방식은 Resolver의 로컬 NS가 Authority NS에서 다수의 IP(라운드 로빈된)를 넘겨받아 캐쉬에 저장해 둔 후 자체적으로도 라운드 로빈처리를 해주지만, 다수의 CNAME 방식은 로컬 NS가 한 개의 주소만을 넘겨받기 때문에 자체 라운드 로빈이 불가능하며, TTL이 만기될 때까지 해당 Resolver들은 하나의 주소를 사용하게 된다.

```
* Authority NS에 직접 질의하였을 경우
$ nslookup  www.nobreak.com  ns.nobreak.com
Name:    www1.nobreak.com
Address:  210.105.79.101
Aliases:  www.nobreak.com

$ nslookup  www.nobreak.com  ns.nobreak.com
Name:    www2.nobreak.com
Address:  210.105.79.102
Aliases:  www.nobreak.com

$ nslookup  www.nobreak.com  ns.nobreak.com
Name:    www3.nobreak.com
Address:  210.105.79.103, 210.105.79.104, 210.105.79.105
Aliases:  www.nobreak.com

* 네임서버의 캐쉬(Third Party Name Server)에서 받아올 경우
$ nslookup  www.nobreak.com  ns.kornet.ne.kr
Name:    www2.nobreak.com
Address:  210.105.79.102
Aliases:  www.nobreak.com

$ nslookup  www.nobreak.com  ns.kornet.ne.kr
Non-authoritative answer:
Name:    www2.nobreak.com
Address:  210.105.79.102
Aliases:  www.nobreak.com

$ sleep 180  (TTL이 만기될 때 까지 기다린 후)

$ nslookup  www.nobreak.com  ns.kornet.ne.kr
Non-authoritative answer:
```

```
Name:    www3.nobreak.com
Address:  210.105.79.103, 210.105.79.104, 210.105.79.105
Aliases:  www.nobreak.com

$ nslookup  www.nobreak.com  ns.kornet.ne.kr
Non-authoritative answer:
Name:    www3.nobreak.com
Address:  210.105.79.104, 210.105.79.105, 210.105.79.103
Aliases:  www.nobreak.com

$ nslookup  www.nobreak.com  ns.kornet.ne.kr
Non-authoritative answer:
Name:    www3.nobreak.com
Address:  210.105.79.105, 210.105.79.103, 210.105.79.104
Aliases:  www.nobreak.com
```

참고로, 로드 밸런싱을 구현하기 위해서는 시스템의 부하에 따라 라우팅을 조정하는 스위치나 클러스터링(Clustering) 솔루션을 통하여야 한다.

6Chapter. DNS와 Sendmail

6.1. MX 알고리즘

MX 레코드는 해당 호스트로 수신되는 편지를 다른 호스트로 라우팅 하도록 한다. 원격 호스트에서 다음과 같이 설정된 도메인, nobreak.com 으로 편지를 송신할 경우에 Sendmail의 동작을 알아보자.

```
nobreak.com.            IN      MX 10   layla.nobreak.com.
                        IN      MX 20   marie.nobreak.com.
                        IN      MX 20   suzi.nobreak.com.
```

1. 로컬호스트가 해당 도메인에 대해 인증을 갖는지 확인한다.
2. layla로 배달을 시도한다.
3. marie 혹은 suzi로 배달을 시도한다.
4. (3)에서 시도되지 않은 호스트로 배달을 시도한다.
5. 자체 큐잉 후, 일정 기간동안 주기적으로 2-4의 과정을 반복한다.

흔히 갖는 잘못된 MX의 이해 중 하나는, layla가 다운되어 marie로 편지가 배달되었을 때, 편지가 marie의 메일박스에 저장 된다는 생각으로, 다수의 MX 레코드 설정을 꺼리는 경우이다. 이렇게 되면, 사용자 계정을 양쪽 호스트에 설정하여 두어야 하며, 사용자는 매번 양쪽의 계정을 모두 확인하여야 한다는 노파심 때문일텐데, 일반적으로 낮은 priority를 갖는 호스트는 큐잉서버로 동작하도록 설정하기 때문에 결국 편지는 하나의 호스트(layla)로 집결 된다. 위의 경우 marie와 suzi의 Sendmail이 다음 조건을 만족하도록 설정되어 있으면 메일 큐잉서버로 동작한다.

1. 해당 도메인(nobreak.com)에 대한 인증을 갖지 않아야 한다.
2. 서버는 해당 호스트로의 메일 릴레이(Relay)를 허용하여야 한다.

인증을 갖지 않아야 한다는 것은 Sendmail의 w 클래스(`sendmail.cw` 혹은 `sendmail.cf`의 Cw)에 nobreak.com 도메인이 없어야 함을 의미하고, 메일 릴레이란 수신되는 편지의 최종 배달지가 자신이 아닐 경우, 즉 인증을 갖지 않을 경우 편지를 해당 호스트로 포워딩함을 뜻한다.

근래의 몇몇 OS 배포판들은 기본적으로 Sendmail이 릴레이를 거부하도록 설정되어 있는데, 이러할 경우 해당 호스트는 큐잉서버로 동작하지 못함에 유의하자. 보통 기관의 SMTP 서버는 타인에 의한 남용을 막기위해 내부 IP(혹은 도메인)에 대해서만 릴레이를 허용하도록 설정하지만, 메일 큐잉 서버의 경우는 본 Feature를 끄거나 해당 호스트를 목적지로 하는 편지에 대해서는 릴레이를 허용하도록(이 말의 의미하는 바를 잘 모르겠다면, 모든 IP에 대해 릴레이를 허용토록 설정하라) 조정하여야 한다. (참고: Allowing controlled SMTP relaying in Sendmail 8.9 (http://www.sendmail.org/tips/relaying.html)) layla의 수신 불능으로 인해 marie로 전달되는 편지는 메일큐에 저장되어 있으면서, 일정 기간(`sendmail.cf`에서 지정된 Timeout.queuereturn=5d 만큼)동안 주기적(Sendmail 구동시 지정된, 일반적으로 30분 -q30m)으로 배달이 시도된다. 따라서 반드시 layla만이 nobreak.com 에 대한 인증을 가져야 한다.

```
* layla의 /etc/sendmail.cw (Solaris: /etc/mail/sendmail.cw)
nobreak.com
```

marie와 suzi가 우선순위가 같다는 점에서 다음과 같은 무한루프 시나리오를 생각해 볼 수 있다. layla가 다운되었을 경우 편지는 marie(혹은 suzi)로 배달될 것이고, 해당 도메인의 큐잉서버인 marie는 다시 MX 리스트의 우선순위에 입각하여 포워딩을 시도할 것이다. 결국 편지는 suzi로 배달될 것이고 suzi 또한 이 과정을 반복한다.

정말 그럴까? 그럴수도 있다. 하지만 Sendmail은 편지를 포워딩 하기전에 해당 도메인의 MX 리스트를 파악하고, 자신의 인증된 도메인(`sendmail.cw`에 기록된)이 포함되어 있는지 확인한다. 만약 포함되어 있다면, 매칭되는 도메인중 가장 높은(다수가 매칭 될 수도 있기 때문에) 우선순위를 갖는 도메인을 포함하여 같거나 적은(MX 값이 큰) 레코드를 MX 리스트에서 제외한다.

```
nobreak.com.            IN      MX 10   layla.nobreak.com.
                        IN   MX 20   marie.nobreak.com. (제외)
                        IN   MX 20   suzi.nobreak.com.  (제외)
```

결국 marie의 Sendmail이 유지하는 MX 리스트에는 layla만이 남게되고, 기대한 바대로 편지는 layla가 정상운영될때 까지 큐잉된다.

```
nobreak.com.            IN      MX 10   layla.nobreak.com.
```

그러나 만약 Sendmail이 자신을 MX 리스트에서 찾지 못하면(`sendmail.cw`에 marie.nobreak.com 이 등록되어 있지 않으면), suzi로의 불필요한 포워딩이 발생할 것이고, 불행히 suzi 또한 같은 상황이라면, 결국 무한루프에 빠질수 있다. Sendmail은 이러한 최악의 상황을 막기위해, 최대 포워딩 횟수 제한(`sendmail.cf`의 MaxHopCount=17, 기본값 25)을 갖고 있으므로, 결국 marie나 suzi중 MaxHopCount에 먼저 걸리는 시스템의 관리자(postmaster)와 송신자에게 다음과 같은 내용의 오류 편지를 보내어줄 것이다.

```
From: Mail Delivery Subsystem <MAILER-DAEMON@marie.nobreak.com>
To: postmaster@marie.nobreak.com
Subject: Returned mail: Too many hops 26 (25 max):...

----- The following addresses had permanent fatal errors -----
<nobreak@nobreak.com>

 ----- Transcript of session follows -----
554 Too many hops 26 (25 max):...
===========================================================
Received: from suzi.nobreak.com by marie.nobreak.com ...
Received: from marie.nobreak.com by suzi.nobreak.com ...
...
```

```
Received: from suzi.nobreak.com by marie.nobreak.com ...
Received: from marie.nobreak.com by suzi.nobreak.com ...
...
```

다음은 단일 큐잉 시스템에서 위와같은 루프가 발생하였을 경우의 메일 메시지이다.

```
nobreak.com.            IN      MX 10   layla.nobreak.com.
                        IN      MX 20   marie.nobreak.com.

From: Mail Delivery Subsystem <MAILER-DAEMON@marie.nobreak.com>
To: postmaster@marie.nobreak.com
Subject: Returned mail: Local configuration error

----- The following addresses had permanent fatal errors -----
<nobreak@nobreak.com>

 ----- Transcript of session follows -----
553 marie.nobreak.com. config error: mail loops back to me (MX problem?)
554 <nobreak@nobreak.com>... Local configuration error
============================================================
...
```

DNS에 등록되어 포인팅되는 호스트명을 `sendmail.cw`에 나열하지 않았을 경우 발생할 문제에 대해 확신할 수 없다면, alias된 모든 호스트명을 `sendmail.cw` 파일에 포함하기 바란다.

6.2. MX 레코드를 갖지 않는 호스트

호스트가 편지를 수신한다면 MX 레코드를 갖는 것이 좋다.

```
mailhub         IN      A       210.105.79.107
```

호스트 mailhub는 수신되는 편지를 다른 호스트로 라우팅할 필요가 없기 때문에, 관리자는 MX 레코드를 설정하지 않았다. 물론 이것이 오류는 아니지만 이러한 경우 인터넷상의 한 MTA가 mailhub로 편지를 송신할때 불필요한 트래픽 낭비가 발생할 수 있다.

Sendmail의 경우를 살펴보자. 먼저 Sendmail은 로컬 네임서버를 통해 mailhub를 lookup(type=any) 한다. Authoritative 혹은 Non-authoritative Answer로 A 레코드만을 돌려받게 될텐데, Authoritative Answer의 경우 MX 레코드가 없을지라도 결과를 신뢰하여 추가 질의 없이 해당 IP로 접속을 시도한다. 하지만 MX 레코드가 없는 Non-authoritative Answer에 대해 Sendmail은 명시적으로 MX 레코드를 재 검색한다. 물론 mailhub는 MX 레코드를 갖지 않기 때문에 복귀값은 없고 불필요한 질의만을 수행하였을 뿐이다. 편지를 배달한다는 점에는 두 경우 모두 전혀 문제 없지만, 후자의 경우와 같은 낭비를 막기 위해서 mailhub는 다음과 같이 설정되는 것이 바람직하다.

```
mailhub         IN      A       210.105.79.107
                IN      MX 10   mailhub.nobreak.com.
```

mailhub 자신이 최종 편지 배달지이기 때문에, 자신을 MX로 설정했다. 이렇듯, 편지를 수신하는 모든 호스트는 비록 자기 자신일지라도 한개 이상의 MX 레코드를 갖는 것이 좋다.

6.3. MX와 CNAME의 연동

참고: RFC974 (ftp://ftp.openbird.com/pub/rfc/rfc974.txt)

MX 레코드를 CNAME과 연결하는 것은 매우 안 좋은 생각이다. 사실 관련 RFC 문서 어디에도 아직 MX 레코드와 CNAME의 연동에 대해 주의 수준을 넘어 명확히 제한을 둔 글이 없기 때문에 이러한 사용이 반드시 안 된다고 말할 수는 없지만, 다음과 같이 불필요한 추가 쿼리로 인해 발생되는 트래픽을 사유로 오랜 기간 잘못된 사용으로 인식되어 왔고, 대부분의 관련 문서에서는 같은 연유로 강력히 사용치 말기를 권한다.

```
layla           IN      MX 10   marie.nobreak.com.
marie           IN      CNAME   suzi.nobreak.com.
suzi            IN      CNAME   mailhub.nobreak.com.
mailhub         IN      A       210.105.79.75
                IN      MX 10   mailhub.nobreak.com.
```

layla로 편지를 배달하기 위한 lookup 과정은 다음과 같다.

1. lookup layla.nobreak.com -> return marie.nobreak.com
2. lookup marie.nobreak.com -> return suzi.nobreak.com
3. lookup suzi.nobreak.com -> return mailhub.nobreak.com
4. lookup mailhub.nobreak.com -> return 210.105.79.75

느슨한 표준으로 인해 각각의 응용마다 조금씩 구현이 다른데, 대표적으로 BIND의 경우는 이를 잘못된 사용("layla.nobreak.com" IN MX points to a CNAME marie.nobreak.com)이라 Syslog를 통해 알려주시만, 몇몇 MTA와 Sendmail는 이를 수용(권장하지는 않지만)하고 있다. 하지만, 몇몇 MTA가 해당 시스템으로 편지를 배달하지 못하는 잠재적 문제를 안고있기 때문에, MX와 CNAME의 연동은 여전히 좋지 않고, 다음과 같이 고쳐져야 한다.

```
layla           IN      MX 10   mailhub.nobreak.com.
marie           IN      CNAME   mailhub.nobreak.com.
suzi            IN      CNAME   mailhub.nobreak.com.
mailhub         IN      A       210.105.79.75
                IN      MX 10   mailhub.nobreak.com.
```

또한, 편지 송신전 Sendmail은 수신인의 주소에 포함된 CNAME을 실제 호스트명으로 수정한후 배달함에 유의하자. 즉 user@marie.nobreak.com 로 보내어지는 편지는, user@mailhub.nobreak.com 으로 번역된 후 송신되며, 이는 수신인의 편지 헤더에 표시되는 To: 정보가 user@marie.nobreak.com 이 아닌 user@mailhub.nobreak.com 임을 뜻한다. 중요한 것은 이러한 변환작업이 송신전에 이루어진다는 점이며, 이는 혼란을 야기할 수 있고, 추후 설명할 Sendmail의 Virtual Domain 설정에서 기대하지 않은 매핑 문제를 발생시킬 수도 있다. 따라서

CNAME으로 지정된 호스트는 메일주소로 사용하지 말거나, CNAME 대신 A 레코드와 MX 레코드를 사용하는 것이 좋다.

6.4. Nonrecursive MX lookup

다음과 같은 경우, mailhub로 배달되는 편지를 고려해보자. 일반적으로 편지가 mx2로 배달되리라 기대할 수 있지만, Sendmail은 MX 레코드에대해 Nonrecursive 하기 때문에, 실제 편지는 mx1(반드시 A 레코드를 갖어야 한다)으로 배달된다.

```
mailhub         IN      MX 10   mx1.nobreak.com.
mx1             IN      MX 10   mx2.nobreak.com.
                IN      A       210.105.79.231
```

만약 Sendmail이 MX 레코드를 CNAME같이 Recursive하게 좇아간다면, 다음과 같은 잠재적인 루프를 허용하게 되므로, Sendmail은 MX 레코드를 좇아가지 않는다.

```
mailhub         IN      MX 10   mx1.nobreak.com.
mx1             IN      MX 10   mx2.nobreak.com.
mx2             IN      MX 10   mailhub.nobreak.com.    ; Loop
```

6.5. Sendmail의 Virtual Domain 설정

하나의 호스트에 연결된 도메인 nobreak.com 과 kr.freebsd.org 에 대해, 일반적으로 nobreak@nobreak.com 과 nobreak@kr.freebsd.org 의 수신자는 같다. 여기에서는 각각의 도메인별로 aliases 테이블을 갖는 것과 같이, 도메인에 따라 수신되는 편지를 다르게 처리하는 방법에 대해 알아본다. 본 글이 Sendmail을 주제로 하지는 않기에, Sendmail을 어느 정도 알고 있다는 가정으로 골자내용만을 다루도록 하겠다. Sendmail에 익숙치 않다면 내용을 이해하기 위해 RuleSet, m4, aliases, dbm/btree/hash 등에 대한 사전학습이 필요할 수도 있다.

설명하고자 하는 방법은 Sendmail 8.8 이상에서 가능하므로,먼저 다음과 같이 설치된 버젼을 확인한다.

```
$ telnet localhost smtp
Trying 127.0.0.1...
220 kfug2.kr.freebsd.org ESMTP Sendmail 8.9.3/8.9.3
```

버젼이 8.8보다 낮다면, ftp://ftp.sendmail.org/pub/sendmail/ 에서 최신버젼을 구하기 바란다. Sendmail 8.2 이상에서는 domaintable Feature를 사용해 구현될 수 있지만, 여기서 제시하는 Sendmail 8.8 이상의 virtusertable Feature로 대체되는 추세이므로 여기에서는 다루지 않겠다.

6.5.1. m4를 사용한 sendmail.cf 생성

먼저 `sendmail-VERSION/cf/cf/`에서 시스템에 적절한 mc 파일을 복사한 후, Virtual Domain 설정에 필요한 virtusertable, genericstable Feature를 추가한다. 작성된 mc 파일은 일반적으로 다음과 같이 보인다.

```
* generic-bsd4.4.mc 파일을 사용해 작성한 freebsd.mc 파일
divert(0)dnl
VERSIONID(`@(#)generic-bsd4.4.mc        8.7 (Berkeley) 5/19/1998')
OSTYPE(bsd4.4)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
FEATURE(virtusertable, btree /etc/domainaliases.db)dnl
FEATURE(genericstable, btree /etc/useraliases.db)dnl
```

virtusertable Feature는 수신(in-bound)되는 편지를 로컬 사용자로 매핑하는 RuleSet을 생성하고, genericstable Feature는 반대로 송신(out-bound)되는 편지에 대해 로컬 유저에 대한 메일주소(메일 헤더상의 From: 필드)를 매핑하는 RuleSet을 생성한다. DNS의 Forward Zone과 Reverse Zone을 생각하면 되겠다.

btree 는 해당 테이블(domainaliases.db)이 B-Tree(Berkeley DB) 데이터베이스 형태로 저장됨을 의미한다. 가능한 다른 유형으로는 hash(Berkeley DB)와 dbm(NDBM)이 있는데, 테이블이 방대할 경우 효율은 btree(Berkeley DB) > hash(Berkeley DB) > dbm(NDBM) 순이다. 속도가 빠른 btree와 hash를 사용하기 위해서는 Berkeley DB(FreeBSD를 비롯해 몇몇 Unix에는 기본으로 포함된다)가 설치되어 있어야 하며, `sendmail`과 `makemap` 컴파일시 -DNEWDB 플래그(Berkeley DB가 설치되어 있다면 자동으로 -DNEWDB 플래그가 사용된다)를 사용해야 한다.

Berkeley DB가 설치되어 있지 않을 경우엔 -DNDBM 플래그(NDBM 라이브러리는 대부분의 Unix에 기본으로 포함된다)로 컴파일 되는데 이때에는 DBM을 사용하여야 한다. 리눅스와 같이 Berkeley DB의 GNU판인 GDBM이 사용되는 경우엔 별도의 노력없이 세 종류의 DB를 모두 이용할 수 있다.

작성된 `freebsd.mc` 파일은 다음과 같이 m4(Macro Language Processor)를 통해 `sendmail.cf`로 변환한다.

```
# cd sendmail-VERSION/cf/cf
# m4 ../m4/cf.m4 freebsd.mc > freebsd.cf
# cp freebsd.cf /etc/sendmail.cf
```

만약 기존의 `sendmail.cf`가 유지되어야 하는 상황이라면, 실제 본 두 Feature가 생성하는 다음의 RuleSet을 참고하여 `sendmail.cf`를 직접 수정하여도 좋다. (RuleSet이 삽입되는 위치를 파악하기 위해 m4로 `sendmail.cf`를 생성하여 비교하기 바란다)

```
* FEATURE(virtusertable, btree /etc/domainaliases.db)dnl
# Virtual user table (maps incoming users)
Kvirtuser btree /etc/domainaliases.db

# handle virtual users (RuleSet 0의 Parse1 아래에 위치)
```

```
R$+ < @ $=w . >         $: < $(vir-
tuser $1 @ $2 $@ $1 $: @ $) > $1 < @ $2 . >
R<@> $+ + $* < @ $* . >
                        $: < $(vir-
tuser $1 + * @ $3 $@ $1 $: @ $) > $1 + $2 < @ $3 . >
R<@> $+ + $* < @ $* . >
                        $: < $(vir-
tuser $1 @ $3 $@ $1 $: @ $) > $1 + $2 < @ $3 . >
R<@> $+ < @ $+ . >      $: < $(virtuser @ $2 $@ $1 $: @ $) > $1 < @ $2 . >
R<@> $+                 $: $1
R< error : $- $+ > $*   $#error $@ $(dequote $1 $) $: $2
R< $+ > $+ < @ $+ >     $: $>97 $1

* FEATURE(genericstable, btree /etc/useraliases.db)dnl
# Generics table (mapping outgoing addresses)
Kgenerics btree /etc/useraliases.db

# handle generics database (RuleSet 93 아래에 위치)
R$+ < @ $=G . > $: < $1@$2 > $1 < @ $2 . > @    mark
R$+ < @ *LOCAL* >       $: < $1@$j > $1 < @ *LOCAL* > @ mark
R< $+ > $+ < $* > @     $: < $(generics $1 $: $) > $2 < $3 >
R< > $+ < @ $+ >        $: < $(generics $1 $: $) > $1 < @ $2 >
R< $* @ $* > $* < $* >  $@ $>3 $1 @ $2                  found qualified
R< $+ > $* < $* >       $: $>3 $1 @ *LOCAL*             found unqualified
R< > $*                 $: $1                           not found
```

6.5.2. Forward-map 테이블 작성

먼저, 해당 호스트가 수신하는 도메인을 클래스 w(`sendmail.cw`)에 나열하여야 한다.

```
* /etc/sendmail.cw 파일
nobreak.com
kr.freebsd.org
```

domainaliases 파일을 편집기로 열어 다음과 같이 Virtual User Table을 작성한다.

```
* /etc/domainaliases 파일
webmaster@nobreak.com           dipper
moonhunt@nobreak.com            moonhunt@ieee.org
cjh@nobreak.com                 error:nouser Unknown User
@nobreak.com                    nobreak
@kr.freebsd.org                 %1@xfree86.org
```

메일주소 webmaster@nobreak.com 는 로컬 사용자 dipper로 매핑되고, moonhunt@nobreak.com 로 배달되는 편지는 moonhunt@ieee.org 로 포워딩 된다. cjh@nobreak.com 으로 배달되는 편지에 대해서는 에러메시지 'Unknown User'로 답하며, 그외 nobreak.com 으로 수신되는 모든 편지는 로컬 사용자 nobreak로 전달된다. 또한, kr.freebsd.org

로 수신되는 모든 편지는 xfree86.org 의 같은 사용자에게 포워딩된다. 다음과 같이 aliases 파일과 연동하여 다수의 사용자를 매핑하는것 또한 가능하다.

```
* /etc/domainaliases 파일
announce@kr.freebsd.org         announce.ml
admin@kr.freebsd.org            admin.ml

* /etc/aliases 파일
announce.ml:            :include:/var/ml/announce.ml
admin.ml:               nobreak, moonhunt@ieee.org
```

6.5.3. Reverse-map 테이블 작성

호스트명이 kfug2.kr.freebsd.org 일때, 시스템에서 out-bound되는 편지는 기본적으로 user@kfug2.kr.freebsd.org 를 From으로 갖는다. 다음의 Reverse-map 테이블은 로컬 사용자의 메일주소를 변경한다.

```
* /etc/useraliases 파일
nobreak                 nobreak@nobreak.com
cjh                     cjh@kr.freebsd.org
```

6.5.4. 테이블 변환 및 실험

작성된 테이블은 makemap을 사용해 dbm/btree/hash 형태로 변환되어야 한다. 여기서는 btree를 사용하였으므로 다음과 같이한다.

```
# makemap btree /etc/domainaliases.db < /etc/domainaliases
# makemap btree /etc/useraliases.db < /etc/useraliases
# ls /etc/domainaliases* /etc/useraliases*
/etc/domainaliases      /etc/useraliases
/etc/domainaliases.db   /etc/useraliases.db
```

이제 Sendmail을 행업하고 동작을 확인하자.

```
# kill -HUP `cat /var/run/sendmail.pid`

# sendmail -bv webmaster@nobreak.com
webmaster@nobreak.com... deliverable: mailer local, user dipper

# sendmail -bv cjh@nobreak.com
cjh@nobreak.com... Unknown User

# sendmail -bv anybody@kr.freebsd.org
anybody@kr.freebsd.org... deliverable: mailer esmtp,
host xfree86.org., user anybody@xfree86.org
```

7Chapter. BIND 유지/보수

7.1. BIND 시그널

BIND는 다음과 같이 몇 개의 예약된 시그널을 사용한다. 시그널 중 몇몇에 대해서는 파일로 결과를 출력하는데, 파일은 보통 `/var/named/` 혹은 `/var/tmp/` 디렉토리에 남는다.

HUP

BIND를 재시작 한다. 보통 부트 파일이나 존 데이터베이스를 수정한 후, 본 시그널을 사용한다. 하지만 이는 BIND를 종료한후, 재시작하는 것과는 다르게, 부트 파일과 수정된 Zone 데이터베이스(Serial이 증가한)만을 업데이트하고, 캐쉬를 유지한다.

INT

내부적으로 BIND는 루트 캐쉬와 존 데이터베이스들을 $ORIGIN으로 분리한 하나의 목록으로 관리하는데, 이 내부 데이터베이스를 `named_dump.db` 파일로 저장한다.

ILL(BIND-8)/IOT(BIND-4)

BIND의 통계정보를 `named.stats` 파일로 생성한다. 네임서버 유지, 관리에 필요한 여러 통계 자료가 들어있다.

USR1

디버깅 정보를 `named.run` 파일로 출력한다. BIND는 본 시그널을 받을 때마다 디버깅 레벨을 한 단계씩 증가시키는데, 각 레벨마다 표시하는 정보의 유형이 조금씩 상이하며, 일반적으로 레벨이 높을수록 보다 자세한 정보가 기록된다.

USR2

디버깅 출력을 종료한다.

WINCH

BIND는 기본적으로 몇몇 주요 메시지만을 Syslog에 남긴다. 본 시그널은 토글 형식으로 작동하며, 요청되는 모든 쿼리를 Syslog에 남기도록 한다.

7.2. BIND Syslog 메시지들

BIND는 Syslog를 통해 일상적 알림에서부터 치명적 오류까지 다양한 메시지를 남긴다. 네임서버의 일반적인 오류 및 오동작의 원인은 Syslog에서부터 찾아나가는 것이 순서이겠다. Syslog는 /etc/syslog.conf의 설정에 따라 다르지만, 보통 /var/log/messages 혹은 /var/adm/messages 파일에 메시지를 남긴다.

다음은, BIND가 구동될 때, HUP 시그널을 받았을 때, 종료될 때 기록되는 일상적인 메시지들이다.

```
Jan  1  5:33:39 ns named[288]: starting.  named 8.2
Jan  2 16:57:11 ns named[291]: reloading nameserver
Feb  5  3:34:31 ns named[289]: named shutting down
```

Syslog에 남는 메시지의 형식은 ’시간 호스트명 named[PID]: 메시지’와 같은 형식를 취하는데, BIND-4와 BIND-8 그리고 각 버전별로 사용되는 단어와 메시지 양식이 조금씩 상이함에 유의한다.

```
Feb  1 11:33:40 ns named[288]: master zone "nobreak.com"
(IN) loaded (serial 1999012901)
```

Zone 데이터베이스를 메모리에 적재하였음을 의미한다. BIND-4에서는 ’primary zone’이라 표현하였지만, BIND-8로 넘어오면서 ’master zone’으로 명칭이 수정되었다.

매시간 BIND는 간략한 통계정보를 기록한다. (BIND-4의 일부 버전과 몇몇 OS 벤더가 제공하는 BIND는 이 Feature를 기본으로 꺼놓고 있다)

```
Feb  2 18:57:12 ns named[291]: USAGE 917949432 917837292
CPU=1558.86u/491.54s CHILDCPU=921.29u/1228.57s

Feb  2 18:57:12 ns named[291]: NSTATS 917949432 917837292
Unknown=3 A=95677 NS=3 SOA=275 PTR=563972 MX=11355 TXT=16 AXFR=10 ANY=17840

Feb  2 18:57:12 ns named[291]: XSTATS 917949432 917837292
RR=270905 RNXD=92478 RFwdR=133260 RDupR=505 RFail=8882
RFErr=0 RErr=75 RAXFR=10 RLame=111437 ROpts=0
SSysQ=24028 SAns=467186 SFwdQ=221971 SDupQ=255265 SErr=5319
RQ=689151 RIQ=17 RFwdQ=0 RDupQ=224 RTCP=53 SFwdR=133260
SFail=290 SFErr=0 SNaAns=189120 SNXD=186481
```

각 메시지의 처음에 나오는 2개 숫자는 현재시간과 BIND가 시작된 시간을 의미한다. ’917949432 - 917837292’를 계산하면, 서버가 운용된 시간(초)을 알 수 있다. USAGE는 CPU 사용정도를 나타내는데, 주 CPU가 1558초동안 사용자(u) 모드에서, 491초동안 시스템(s) 모드에서 동작하였음을 알 수 있다. CHILDCPU도 같은 의미인데, 멀티 프로세스 시스템이 아니라면, CHILDCPU는 0u/0s 로 표시될 것이다. NSTATS와 XSTATS는 BIND Statistics에서 자세히 다룬다.

```
Feb  5 09:39:05 ns named[26674]: Sent NOTIFY for "nobreak.com IN SOA"
(nobreak.com); 1 NS, 1 A
```

BIND-8에 포함된 Dynamic Update 기능은 Primary가 Secondary에게 Zone 데이터베이스가 수정되었으니 업데이트하라는 정보를 보낸다. 본 메시지는 nobreak.com 에 대한 Zone 데이터베이스가 업데이트되어 알림 메시지를 보냈다는 뜻이다.

```
Feb  5 09:36:00 ns named[26674]: ns_forw: query(57.132.119.210.in-
addr.arpa)
```

```
No possible A RRs
```

Resolver가 요청한 도메인(인버스 도메인)을 찾을 수 없을 경우 이다.

```
Feb  5 09:35:46 ns named[26674]: Lame server on 'ryu.siyon.com'
(in 'siyon.com'?): [210.105.79.103].53 'ns.mylover.com'
```

도메인 siyon.com 이 ns.mylover.com 으로 위임되었으나, 해당 네임서버가 Authority 설정이 되어 있지 않은 경우이다. 이것은 외부 네트워크에서의 Lame Delegation이므로 신경 쓰지 않아도 좋다.

7.3. BIND Syslog 주의/오류 메시지들

다음의 메시지들은 네임스페이스상의 비정상적인 링크와, 잘못된 네임서버 설정에 기인한 메시지들이다. 본 메시지 중 몇몇은 타 네임서버에 의한 것이고, 의도적인 경우도 있으니, 가능한 범위에서 원인을 제거하도록 한다.

```
Feb  3 12:53:27 ns named[291]: ns_forw: query(shpark.co.kr)
contains our address (ns.nobreak.com:210.105.79.2)
```

도메인 shpark.co.kr 이 ns.nobreak.com(자신)으로 위임되었으나, Authority가 설정되어 있지 않을 경우이다. 내부 네트워크에 대한 Lame Delegation 메시지이니, 해당 도메인을 확인하고 적절한 조치를 취하도록 한다.

```
Feb  5 10:09:01 ns named[26674]: "nobreak.com IN NS"
points to a CNAME (ns.nobreak.com)

Feb  5 10:09:11 ns named[26674]: "nms.nobreak.com IN MX"
points to a CNAME (mail.nobreak.com)

Feb  5 09:57:15 ns named[26674]: power.nobreak.com has CNAME and other data
```

CNAME의 잘못된 사용에 기인한 오류들이다. 특히 MX와 관련된 오류는 전체가 아닌 몇몇 MTA(예:sendmail)에서 메일 라우팅에 문제가 생길 수 있기 때문에, 원인을 찾기위해 오랜 시간을 허비할 수 있으므로 주의하자. 다음과 같은 문법적 오류가 있을 때, 본 메시지들이 나타난다.

```
nobreak.com.     IN      NS      ns.nobreak.com.   ; (X) ns는 CNAME이다.
ns               IN      CNAME   mild.nobreak.com.

nms              IN      MX      mail.nobreak.com. ; (X) mail은 CNAME 이다.
mail             IN      CNAME   mhub.nobreak.com.

power            IN      CNAME   layla.nobreak.com.
                 IN      MX      mhub.nobreak.com. ; (X)추가레코드가 올 수 없
다.
```

CNAME 레코드는 어떠한 추가 레코드도 갖을 수 없으며, NS/MX/SOA 레코드는 CNAME과 연결될 수 없음을 기억하자.

```
Feb  5 09:24:56 ns2 named-xfer[2502]: record too short from
[210.105.79.2], zone nobreak.com
```

Primary NS가 xfrnets(BIND-4) 혹은 allow-transfer(BIND-8) 옵션으로, Zone Transfer를 막아, 해당 Zone을 갖고 오지 못할 경우이다. 해당 네임서버 관리자에게 연락하여, Zone Transfer가 가능하도록 하여야 한다.

7.4. BIND Statistics

네임서버 활용정도와 도메인내에 요구되는 로컬 네임서버 개수를 파악하기 위하여, 주기적인 통계 자료 검토가 필요하다. 여기서 네임서버 통계 정보 분석에 대해 알아보고자 한다. 통계정보는 [그림 6]과 같이 timeout(DNS는 기본적으로 UDP를 사용하기 때문에)에 의한 중복된 쿼리를 포함한다.

Figure 7-1. 네임서버간의 질의 예제

BIND로부터 통계 정보를 얻기 위해서는 다음과 같이 ILL(BIND-8)/IOT(BIND-4) 시그널을 사용한다. BIND는 시그널을 받으면 통계 파일을 `/var/named/named.stats` 혹은 `/var/tmp/named.stats`로 출력한다.

```
# kill -ILL `cat /var/run/named.pid`
# ls -al /var/named/named.stats
-rw-r--r--
1 root     root        19832 Mar  8 04:51 /var/named/named.stats
# cat /var/named/named.stats
+++ Statistics Dump +++ (920978301) Tue Mar  9 20:18:21 1999
208719  time since boot (secs)
3396    time since reset (secs)
3       Unknown query types
205739  A queries
39      NS queries
916     SOA queries
802719  PTR queries
19192   MX queries
2       TXT queries
15      AAAA queries
```

```
129     AXFR queries
29359   ANY queries

++ Name Server Statistics ++
(Legend)
        RR      RNXD    RFwdR   RDupR   RFail
        RFErr   RErr    RAXFR   RLame   ROpts
        SSysQ   SAns    SFwdQ   SDupQ   SErr
        RQ      RIQ     RFwdQ   RDupQ   RTCP
        SFwdR   SFail   SFErr   SNaAns  SNXD
(Global)
        447781  164719 227112   869     25991
        0       245    125      170586  0
        33963   720395 337570   294284  8122
        1058113 0      337570   190     0
        227112  20791  0        228616  298214

[210.104.247.129]
        157     0      0        0       10
        0       0      0        147     0
        0       36     146      22      0
        36      0      0        0       0
        11      0      0        0       13

.
.
.
```

위는 BIND 8.2에서 통계정보를 출력한 예이다.

```
208719  time since boot (secs)
3396    time since reset (secs)
```

BIND가 구동된후 운용된 시간과 마지막으로 리로드된(HUP 시그널을 받은) 후 경과된 시간을 초단위로 표시한다.

```
3       Unknown query types
```

알려지지 않은 쿼리에 대한 질의 횟수이다. 이는 잘못된 구현에 기인하거나, 누군가에 의한 새로운 타입시도 때문이다.

```
205739  A queries
```

A 쿼리는 대부분의 응용에서 요구하는 질의이며, 가장 빈번히 요구된다.

```
39      NS queries
```

내부적으로 BIND는 루트 서버에 질의 할때, NS 쿼리를 사용한다. 해당 서버가 루트 서버가 아닐 경우에는 Dig나 Nslookup같은 질의 도구에 의한 명시적 요청을 뜻한다.

```
916     SOA queries
```

SOA 쿼리는 Secondary NS가 해당 Zone의 시리얼 변화를 감지하기 위해 사용한다.

```
802719  PTR queries
```

Reverse 도메인을 요구하는 응용들에 의하며 A 쿼리와 함께 가장 빈번히 요청된다.

```
19192   MX queries
```

MX 쿼리는 Sendmail과 같은 MTA가 메일 라우팅 정보를 습득하기 위해 요청한다.

```
2       TXT queries
15      AAAA queries
```

TXT와 AAAA 쿼리는 Dig나 Nslookup같은 DNS 질의 도구에 의해 요청된다.

```
129     AXFR queries
```

AXFR 쿼리는 Secondary가 Zone Transfer 할 때 요청되므로, 그 수치는 Zone Transfer 횟수를 의미한다.

```
29359   ANY queries
```

Any 쿼리는 근래의 Sendmail이 목적지 호스트의 A, MX, CNAME 정보를 얻기 위해 사용한다.

나머지 통계정보는 개별 호스트(Remote 네임서버와 Stub Resolver)의 통계 정보를 나타낸다. 보통 총 합을 표시하는 [Global] 필드 아래로 수백 혹은 수천의 호스트가 나열되는데, 개별 서버와의 통계정보는 메모리를 소비할 뿐 일반적으로 중요치 않아, BIND-8에서는 기본으로 제거되어 [Global] 필드만이 표시된다. 하지만, 개별 호스트별 통계는 송/수신 패킷에 대한 자세한 내역을 알려주기 때문에, 일반적인 통계에서 파악할 수 없는 문제(네트워크 지연과 같은)를 진단하는데 도움이 되기도 한다. 다음과 같이 부트 파일 옵션을 조정함으로써 개별 호스트 통계를 가능하게 할 수 있다.

```
* named.conf (BIND-8)
options {
        host-statistics yes;
};
```

Legend로 표시되는 구분자들은 각 필드의 의미를 나타내는데, S(Sent)로 시작하는것은 로컬 호스트에서 송신된 쿼리를 의미하고, R(Received)은 수신을 뜻한다. 순서와 종류는 BIND의 버젼에 따라 조금씩 상이할 수 있다.

```
RR(Sent us an answer) 447781
RQ(Sent us a query) 1058113
```

리모트 호스트가 로컬 네임서버로 응답(Answer)한 횟수가 RR이며, 질의(Question)한 횟수가 RQ이다. RR이 RQ에 대한 응답은 아니므로, RR과 RQ의 수치엔 상관관계가 없다.

```
RNXD(Sent us a negative response) 164719
SNXD(Sent them a negative response) 298214
```

RNXD는 요청한 쿼리에 대해 '도메인 없음' 응답을 받았을 경우 증가한다. SNXD는 반대로 '도메인 없음' 쿼리를 전송했을 경우이다.

```
RFwdR(Sent us a response we had to fwd) 227112
SFwdR(Forwarded a response to them) 227112
RFwdQ(Sent us a query we had to forward) 337570
SFwdQ(Forwarded a query to them) 337570
```

Resolver의 요청에 대해 로컬 네임서버는 네임스페이스를 검색하여 최종적으로 리모트 네임서버로부터 응답받은 결과를 Resolver에게 통지하는데, RFwdR은 리모트 네임서버로부터 응답받은 쿼리(RR)중 포워딩할 쿼리의 개수이고, SFwdR은 실제 포워딩한 쿼리 개수이다. 반대로 RFwdQ는 Resolver의 질의(RQ)에 대한 포워딩 요청이며, SFwdQ 실제 리모트 네임서버로 질의를 포워딩한 경우이다.

```
RDupR(Sent us an extra answer) 869
RDupQ(Sent us a retry) 190
SDupQ(Sent them a retry) 294284
```

네임서버가 다운되어 Timeout이 야기되거나, 네트워크 장애 등의 요소로 호스트간 패킷 송/수신에 지연이 발생할 경우, 호스트간에 중복된(Retry) 쿼리 요청이 발생할 수 있다. RDupR은 리모트 호스트에서 로컬 네임서버로 전송한 중복된 응답([그림 6]의 시나리오 참고) 횟수이며, RDupQ는 로컬 네임서버가 수신한 중복 질의(해당 질의를 미처 처리하지 못한 상태에서 수신되는 동일한 질의) 횟수이고, SDupQ는 로컬 네임서버가 리모트 호스트로 요청한 중복 질의 개수이다.

```
RFail(Sent us a SERVFAIL) 5176
SFail(Sent them a SERVFAIL) 90
```

RFail은 호스트(Remote)의 잘못된 Zone 데이터베이스 설정, 메모리 할당 오류, Secondary일 경우 Expire된 도메인 등의 문제로 기인한 SERVFAIL 응답 횟수를 나타낸다. SFail은 로컬 네임서버의 문제로 발송된 SERVFAIL 메시지이다.

```
RFErr(Sent us a FORMERR) 0
SFErr(Sent them a FORMERR) 0
```

RFErr은 수신한 FORMERR 응답 횟수를 나타낸다. FORMERR 응답은 리모트 네임서버가 문법적 오류가 있는 질의를 받았을 때 보내어진다. SFErr은 반대로 로컬 네임서버가 송신한 FORMERR 메시지의 개수이다. 네트워크상의 패킷전송에 문제가 발생하지 않는 한 본 두 값은 0 이다.

```
RErr(Sent us some other error) 245
SErr(Sent failed) 8122
```

수신된 SERVFAIL 과 FORMERR 이외의 모든 수신/송신 오류 메시지는 RErr/SErr에 포함된다.

```
RAXFR(Sent us an AXFR) 125
```

AXFR 쿼리는 Secondary가 Zone Transfer 하기 위해 보내어 진다. 개별 호스트 항목에서 본 필드가 0 일 경우에는 해당 호스트가 로컬 네임서버의 어떠한 도메인에 대해서도 Secondary로 동작하지 않음을 뜻한다.

```
RLame(Sent us a lame delegation) 170586
```

위임된 도메인중 몇몇에대해 Authority가 설정되어있지 않을경우 본 값은 0이 아니다. 즉 특정 도메인의 네임서버로 지정되어 있으나 Primary 혹은 Secondary 설정이 없을 경우이다.

```
ROpts(Sent us some IP options) 0
```

IP 옵션이 설정된 패킷을 수신하였을 경우, ROpts가 증가한다.

```
SSysQ(Sent them a sysquery) 33963
```

로컬 네임서버의 시스템 쿼리 전송횟수이다. 시스템 쿼리는 리모트 네임서버(루트 네임서버를 포함하여) 정보를 업데이트 하기 위해 사용된다.

```
RIQ(Sent us an inverse query) 0
```

RIQ는 수신한 Reserve Domain 요청 횟수이지만, 근래의 BIND에서는 PTR queries로 흡수되어 더 이상 사용되지 않는다. 따라서 RIQ는 항상 0이다.

```
RTCP(Sent us a query using TCP) 0
```

RTCP는 TCP 연결을 통해 수신된 질의 횟수를 의미한다. 대부분의 쿼리는 UDP를 통해 송/수신되므로 본 값은 0을 갖거나, 비율적으로 매우 적은 수치를 유지한다.

```
SAns(Sent them an answer) 720395
SNaAns(Sent them a non-autoritative answer) 228616
```

SAns는 요청된 질의(RQ)에 대한 응답 횟수를 말하며, 결과가 캐쉬에서 발견되었을 경우에는 SNaAns가 카운트된다.

7.5. BIND Debugging Output

BIND의 디버깅 출력은 개발자들이 소프트웨어를 메인터넌스할 목적으로 활용되므로, 또 다른 구현을 생각하거나 BIND의 동작을 파악하기 위한 이유가 아니라면, 굳이 이를 모두 이해하려 할 필요는 없다. 하지만 BIND를 좀더 깊숙이 이해하고 그 응용을 극대화 하고자 한다면, 디버깅 정보 분석에 많은 재미를 느낄 수 있을 것이다. 여기 모두는 아니지만 디버깅 출력 정보를 해석하는 기본적인 아이디어를 소개한다.

```
# kill -USR1 `cat /var/run/named.pid` (or 'ndc trace')
# ping www.openbsd.org.
# kill -USR2 `cat /var/run/named.pid` (or 'ndc notrace')
```

```
# ls -al /var/named/named.run
-rw-r--r--  1 root  bin  1153 Mar 10 00:30 named.run
```

BIND는 USR1 시그널은 받을때마다 디버깅 레벨을 한단계씩 높여가는데, 높은 디버깅 레벨은 좀더 자세한 정보를 표시하여 준다. 디버깅 출력은 `/var/named/named.run` 혹은 `/var/tmp/named.run` 파일로 생성되며, 매우 빠르게 증가하므로 필요한 정보가 잡혔다고 판단되는 시점에서 USR2 시그널을 이용해 출력을 정지시키기 바란다. BIND 디버깅 출력은 다발적으로 발생하는 쿼리에 대한 정보가 모두 기록되므로, 때론 원하는 정보를 추리는데 약간의 인내심이 필요할 수도 있다. BIND의 초기화 과정을 살펴보고자 한다면 **named -d 1 &**과 같이 부팅시 커맨드라인 옵션을 주어야 한다.

```
datagram from [210.105.79.6].3442, fd 21, len 33;
```

이것은 호스트 210.105.79.6 에서 포트 3442번으로 길이 33 byte의 UDP 패킷이 파일 디스크립터 6번을 통해 수신되었음을 뜻한다. 여기서 말하는 파일 디스크립터란 서버 IP 주소(패킷을 listen 하는)에 bound된 소켓 핸들러를 말한다.

```
req: nlookup(www.openbsd.org) id 28375 type=1 class=1
```

요청된 datagram은 www.openbsd.org 에 대한 질의(req)임을 알 수 있다. 구체적으로 클래스 IN(class=1)에 대한 A(type=1) 레코드 요청이며 내부 구분번호는 28375로 매겨졌다.

```
req: missed ’www.openbsd.org’ as ” (cname=0)
```

요청 도메인에 대해 알고있는 자료가 없음(네임서버가 해당 도메인에 대해 Authority를 갖고 있지 않으며, 캐쉬에서도 찾을 수 없을 때)을 뜻한다. cname=0 는 www.openbsd.org 가 CNAME 으로 설정되지 않았음을 말하는데, 물론 지금 단계에서는 BIND가 해당 도메인의 CNAME 설정여부를 알아낼 수 없지만, 본 값이 0이 아닐 경우에는 CNAME이 가르키는 도메인을 대신 찾는다.

```
forw: forw -> [198.41.0.10].53 ds=23 nsid=19461 id=28375 5ms retry 4sec
```

자체 lookup에 실패하였기 때문에 다음 단계로 질의를
’J.ROOT-SERVERS.NET(198.41.0.10:53)’으로 포워딩 한다.

```
datagram from [198.41.0.10].53, fd 21, len 449;
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19461
;; flags: qr; Ques: 1, Ans: 0, Auth: 12, Addit: 12
;; QUESTIONS:
;;      www.openbsd.org, type = A, class = IN

;; AUTHORITY RECORDS:
ORG.    518400  IN      NS      A.ROOT-SERVERS.NET.
ORG.    518400  IN      NS      I.ROOT-SERVERS.NET.
ORG.    518400  IN      NS      K.GTLD-SERVERS.NET.

;; ADDITIONAL RECORDS:
A.ROOT-SERVERS.NET.     3600000 IN      A       198.41.0.4
```

```
I.ROOT-SERVERS.NET.     3600000 IN      A       192.36.148.17
K.GTLD-SERVERS.NET.     518400  IN      A       195.8.99.11
```

J.ROOT-SERVERS.NET 이 요청에 대한 응답을 보내어 왔다. 이처럼 응답이 delegation에 대한 레퍼런싱일 경우 관련 내용이 모두 출력된다. 결과는 캐쉬에 저장된다.

```
resp: nlookup(www.openbsd.org) qtype=1
```

캐쉬에서 www.openbsd.org. 를 다시 찾는다.

```
resp: found 'www.openbsd.org' as 'org' (cname=0)
resp: forw -> [195.8.99.11].53 ds=23 nsid=19462 id=28375 1ms
```

완벽한 결과는 아니지만, ORG 레벨에서 위임정보를 발견하였기 때문에, 'K.GTLD-SERVERS.NET(195.8.99.11)'으로 질의를 포워딩 한다.

```
datagram from [195.8.99.11].53, fd 21, len 219;
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19462
;; flags: qr rd; Ques: 1, Ans: 0, Auth: 5, Addit: 3
;; QUESTIONS:
;;      www.openbsd.org, type = A, class = IN


;; AUTHORITY RECORDS:
OPENBSD.org.    172800  IN      NS      ZEUS.THEOS.COM.
OPENBSD.org.    172800  IN      NS      CVS.OPENBSD.org.
OPENBSD.org.    172800  IN      NS      NS.SIGMASOFT.COM.

;; ADDITIONAL RECORDS:
ZEUS.THEOS.COM.         172800  IN      A       199.185.137.1
CVS.OPENBSD.org.        172800  IN      A       199.185.137.3
NS.SIGMASOFT.COM.       172800  IN      A       209.249.129.35
```

K.GTLD-SERVERS.NET 으로부터 위임정보에 대한 레퍼런싱 응답이 돌아왔다.

```
resp: nlookup(www.openbsd.org) qtype=1
resp: found 'www.openbsd.org' as 'openbsd.org' (cname=0)
```

캐쉬에서 www.openbsd.org. 를 찾는 과정 중, openbsd.org. 레벨의 위임을 발견했다.

```
resp: forw -> [199.185.137.3].53 ds=23 nsid=19467 id=28375 22ms
datagram from [199.185.137.3].53, fd 21, len 272;
```

CVS.OPENBSD.ORG(199.185.137.3) 로 질의를 포워딩한 후, 기대한 응답(A)을 수신하였다. (디버깅 레벨 1에서는 위임 정보만이 표시되기 때문에, 결과의 내용을 보고자 할 경우에는 더 높은 디버깅 레벨을 적용하여야만 한다)

```
send_msg -> [210.105.79.6].3442 (UDP 21) id=28375
```

마지막으로 검색된 결과를 클라이언트에 응답함으로써, 28375 쿼리에 대한 처리가 성공적으로 마무리되었다.

8Chapter. DNS 오류 수정 도구

8.1. NSLOOKUP

네임서버를 운영하고 관리하는데 있어 문제를 발견하고 해결하기 위해 Resolver의 입장으로 네임서버를 시험해볼 필요가 있다. 대부분의 시스템에 기본 설치되어 있는 nslookup은 dig와 함께 가장 널리 사용되는 네임서버 질의 도구로써, 도메인 메니저의 기본 무기중 하나이다.

```
$ nslookup
Default Server:  ns.nobreak.com
Address:  210.105.79.2
> exit
```

nslookup은 실행후 대화형 프롬프트 '>'를 표시하고 `/etc/resolv.conf`에 정의된 첫 번째 네임서버를 기본 질의 서버로 설정한다. **nslookup**은 BIND와 달리 하나의 서버만을 질의에 사용하기 때문에 'Default NS -> Timeout -> Error'와 같이 동작한다.

8.1.1. 도메인 네임 검색

nslookup은 기본적으로 입력된 도메인에 대해 A 레코드를 검색하고, IP 주소(in-addr.arpa)에 대해서는 PTR 레코드를 검색한다. **set type=RR** 설정으로 A 레코드 이외의 레코드 또한 검색할 수 있으며, RR(Resource Record)에는 A, ANY, CNAME, HINFO, MX, NS, PTR, SOA, TXT 등이 올 수 있다. 이중 ANY는 관련된 레코드들을 모두 출력하라는 약속 기호이다.

```
> www.kr.freebsd.org.                   # IP 검색
Name:    www.kr.freebsd.org
Address:  150.183.110.39

> ftp.kr.freebsd.org.
Name:    www.kr.freebsd.org             # ftp는 www의 CNAME
Address:  150.183.110.39
Aliases:  ftp.kr.freebsd.org

> 150.183.110.39                        # 도메인 검색
Name:    www.kr.freebsd.org
Address:  150.183.110.39

> set type=MX                           # MX 레코드 검색
> kr.freebsd.org.
kr.freebsd.org  preference = 10, mail exchanger = mail.kr.freebsd.org

> set type=NS                           # NS 레코드 검색
> kr.freebsd.org.                       # 도메인 위임 확인
kr.freebsd.org     nameserver = ns.kr.freebsd.org
kr.freebsd.org     nameserver = ns2.kr.freebsd.org
```

```
ns.kr.freebsd.org  internet address = 150.183.110.2
ns2.kr.freebsd.org internet address = 150.183.110.3

> 46.102.39.in-addr.arpa.             # 인버스 도메인 위임 확인
kr.freebsd.org     nameserver = ns.kr.freebsd.org
kr.freebsd.org     nameserver = ns2.kr.freebsd.org
ns.kr.freebsd.org  internet address = 150.183.110.2
ns2.kr.freebsd.org internet address = 150.183.110.3
```

8.1.2. 기본 쿼리 서버 변경

nslookup은 기본적으로 recurse 모드로 동작하기 때문에, 때론 해당 도메인의 Authority를 갖는 특정 네임서버에 직접 질의를 하여 Authoritative 응답(네임서버의 캐쉬에서가 아닌)을 확인 할 필요가 있다. server, lserver 명령으로 기본 질의 서버를 변경 할 수 있다. 두 명령은 주어진 네임서버의 주소(쿼리가 아닌)를 찾을 때 사용할 질의 서버의 차이인데, server 는 현재의 기본 서버를 통하고, lserver 는 시스템 기본 서버(nslookup 구동시 초기 설정되는)를 사용함이 다르다. lserver 명령은 타 네임서버로 스위칭 한 후, 다시 다른 네임서버로 스위칭하려 하는데, 현재의 네임서버가 동작하지 않아 해당 네임서버의 주소를 검색하지 못할 때 사용한다. 다음을 보자.

```
$ nslookup
Default Server:  ns.nobreak.com
Address:  210.105.79.2
```

nslookup 구동시의 기본 서버 ns.nobreak.com 이 lserver 명령에서 주어진 NS의 주소를 찾기위한 질의 서버가 된다.

```
> server ns.jp.freebsd.org.        # 기본 서버 변경
Default Server:  ns.jp.freebsd.org
Address:  199.100.7.25

> server ns.nobreak.com.
*** Can’t find address for server ns.nobreak.com: Non-existent host/domain
```

ns.jp.freebsd.org를 통해 ns.nobreak.com을 찾을 수가 없다. 이때에는 **lserver** 명령으로 시스템 기본 서버를 통해 ns.nobreak.com 의 주소를 검색한다.

```
> lserver ns.nobreak.com.
Default Server:  ns.nobreak.com
Address:  210.105.79.2
```

루트 네임서버를 질의 서버로 하고자 할 때는, 간단히 root 명령을 사용할 수 있다.

```
> root
Default Server:  a.root-servers.net
Address:  198.41.0.4
```

8.1.3. 네임 서버처럼 질의하기

네임서버는 Resolver의 요청을 처리하기 위해, 네임스페이스를 검색하며, 여러 네임서버와 통신을 하는데, nslookup으로 동일한 과정을 밟아보도록 하자. 네임서버가 인터넷상에서 어떻게 동작하며, 네임서버들 간에는 어떤 사건들이 발생하고, 여러분을 위해 무엇을 하는지, 구체적인 느낌을 받을 수 있을 것이다.

Figure 8-1. 네임서버처럼 질의하기

```
(1)
> set norecurse     # Iterative 모드로 전환
> www.kr.freebsd.org.
Server:  ns.nobreak.com
Address:  210.105.79.2

Name:    www.kr.freebsd.org
Served by:
- H.ROOT-SERVERS.NET
          128.63.2.53
          ORG
- B.ROOT-SERVERS.NET
          128.9.0.107
          ORG
...
```

ORG. 가 관리되는 루트 서버들의 목록을 레퍼런싱 해준다.

```
(2)
> server h.root-servers.net.
> www.kr.freebsd.org.
Server:  h.root-servers.net
Address:  128.63.2.53

Name:    www.kr.freebsd.org
Served by:
- WHO.CDROM.COM
          204.216.27.3
          FREEBSD.ORG
- NS1.CRL.COM
          165.113.1.36
          FREEBSD.ORG
- NS2.CRL.COM
```

```
          165.113.61.37
          FREEBSD.ORG

(3)
> server who.cdrom.com.
> www.kr.freebsd.org.
Server:  who.cdrom.com
Address:  204.216.27.3

Name:    www.kr.freebsd.org
Served by:
- ns.kr.freebsd.org
          150.183.110.2
          kr.freebsd.org
- ns2.kr.freebsd.org
          150.183.110.3
          kr.freebsd.org

(4)
> server ns.kr.freebsd.org.
> www.kr.freebsd.org.
Server:  ns.kr.freebsd.org
Address:  150.183.110.2

Name:    www.kr.freebsd.org
Address:  150.183.110.39
```

8.1.4. Zone Transfer

해당 도메인의 Zone에 대한 복사본을 얻기위해, Primary로부터 Zone 데이터베이스를 끌어오는 작업을 Zone Transfer라 한다. 이 작업은 주로 Secondary NS 측에서 이루어지며, 때때로 얼마나 많은 수의 호스트가 등록되어 있는지 혹은 Zone의 문법적 오류를 검사하기 위해 관리자가 수동으로 조작하기도 한다. Zone Transfer는 Authority를 갖는 네임서버에 직접 질의하여야 하므로, nslookup 상에서 해당 NS로 질의 서버를 변경한후, ls 명령을 사용한다.

```
> server ns.kr.freebsd.org.
> ls -t A kr.freebsd.org.      # A 레코드 출력
 kr.freebsd.org.                server = ns.kr.freebsd.org
 kr.freebsd.org.                server = ns2.kr.freebsd.org
 mail                           150.183.110.32
 mqueue                         150.183.110.33
 www                            150.183.110.39
 www2                           150.183.110.40

> ls -d kr.freebsd.org.        # 모든 레코드 출력
 kr.freebsd.org.                SOA   ns.nobreak.com hostmaster.kr.freebsd.org.
                                (1999031501 21600 1800 1209600 86400)
```

```
 kr.freebsd.org.                NS    ns.nobreak.com
 kr.freebsd.org.                NS    ns2.nobreak.com
 kr.freebsd.org.                MX    10   mail.kr.freebsd.org
 kr.freebsd.org.                MX    20   mqueue.kr.freebsd.org
 cvsup                          CNAME www.kr.freebsd.org
 mail                           A     150.183.110.32
 mqueue                         A     150.183.110.33
 ftp                            CNAME www.kr.freebsd.org
 ftp2                           CNAME www2.kr.freebsd.org
 ftp3                           CNAME ftp.free.nobreak.com
 www                            A     150.183.110.39
 www                            HINFO Pentium-200  FreeBSD 2.2.8
 www                            TXT  "Korea FreeBSD Users Group"
 www2                           A     150.183.110.40
 www2                           HINFO Pentium-133MHz  FreeBSD 2.2.8
 www2                           TXT  "Korea FreeBSD Users Group"

> ls -t MX kr.freebsd.org > MX-kr.freebsd.org   # 파일로 저장
> view MX-kr.freebsd.org                        # 파일 내용 확인
```

BIND의 경우 **named-xfer**라는 외부 프로그램을 사용해 Zone Transfer를 수행한다. 네임서버의 입장에서 부트 파일에 Secondary 설정이 있을 경우의 처리과정을 살펴보자.

```
secondary       kr.freebsd.org  210.105.79.2    sec-kr.freebsd.org
```

BIND는 secondary 명령을 만나면 내부적으로 다음과 같이 동작한다.

```
loop(Interval == TTL) {
        named-xfer -z kr.freebsd.org -f /var/named/sec-kr.freebsd.org -
s Current_Serial 210.105.79.2
        switch ( $? ) {      // named-xfer 는 환경 변수 '$?'에 결과를 복귀함
                case 0 : OK; // 시리얼이 같음, Zone Transfer가 필요치 않음
                case 1 : OK; // 시리얼이 증가했음, Zone Transfer가 성공적으
로 수행됨
                case 2 : ERROR; // 네임서버를 찾을 수 없음
                                // 혹은 네임서버가 도메인의 Authority를 갖지 않
음
                case 3 : ERROR: // 시리얼이 감소했음, 기존의 백업카피 유지
        }
        primary  kr.freebsd.org  sec-kr.freebsd.org
}
```

보안의 이유로 허락된 곳(예: Secondary NS's IP)에서만 Zone Transfer를 허용하고자 한다면, Primary NS의 부트파일에 다음과 같은 옵션을 준다. (Zone Transfer만을 제한하는 것이기 때문에, 호스트에 대한 개별 쿼리는 허용된다)

```
xfrnets  210.105.79.3&255.255.255.255  210.105.80.128&255.255.255.128
```

이것은 BIND-4의 설정예인데, IP 210.105.79.3과 210.105.80.129-254 에서만 Zone Transfer를 허용하라는 의미이다. BIND-4에서는 개별 IP와 서브넷으로 나누어진 블럭에 대해 "IP&Mask"의 형식으로 목록을 작성하며, 클래스 전체를 허용하고자 할 경우엔 210.105.79.0과 같이 마스크를 생략하여도 된다.

```
options {
        allow-transfer { localnets; 210.105.79.3; };
};
```

BIND-8의 경우에 해당 서버가 속한 네트워크와, 210.105.79.3만을 허용한 예이다. localnets는 예약어이며 다른 예약어로는 any, none, localhost 가 있다.

8.1.5. 초기화 파일 .nslookuprc

nslookup은 실행시 `~/.nslookuprc` 파일이 존재하면, 내용을 읽어 옵션을 조정한다. 매번 설정하는 옵션이 있다면, 본 파일을 통해 간편화할 수 있겠다.

```
* .nslookuprc 파일 예
set type=NS
set nosearch
set debug
```

8.2. DIG

Dig(Domain Information Groper)의 사용법을 조금만 짚어보도록 하자. **nslookup**과의 기능적 차이는 크게 없지만, 사용이 간결하고, 출력이 상세하여, Shell Script등에서 주로 사용된다. 다음은 ns.kornet.ne.kr을 통해 www.nobreak.com의 A 레코드를 검색한 결과이다.

```
$ dig [@네임서버] 도메인 [쿼리타입] [+쿼리옵션]
$ dig  @ns.kornet.ne.kr  www.nobreak.com  A
;; ANSWER SECTION:
www.nobreak.com.        16h12m36s IN CNAME  ns.nobreak.com.
ns.nobreak.com.         1d19h12m27s IN A  210.105.79.2
;; AUTHORITY SECTION:
nobreak.com.            22h17m35s IN NS  ns.nobreak.com.
nobreak.com.            22h17m35s IN NS  ns2.nobreak.com.
;; ADDITIONAL SECTION:
ns.nobreak.com.         1d19h12m27s IN A  210.105.79.2
ns2.nobreak.com.        1d1h46m58s IN A  210.105.79.3
```

DIG는 쿼리에 대한 결과를 ANSWER SECTION에, 해당 도메인의 인증을 갖는 네임서버 정보를 AUTHORITY SECTION에, 그리고, 글루레코드 등이 있을 경우 그에대한 정보를 ADDITIONAL SECTION에 출력하여 준다.

8.3. DNSWALK

Dnswalk는 Perl로 제작된 스크립트로써 존재하지 않는 호스트를 가르키는 MX, PTR이 없는 A, 잘못된 CNAME, 유효하지 않은 이름문자, 누락된 트레일링 도트, 불필요나 글루 레코드, Lame Delegation 등 Zone 데이터베이스의 다양한 오류를 찾아준다. 미처 발견치 못한 문제점을 진단하는데 도움이 될 것이다. Dnswalk의 최신 버전은 다음 주소에서 얻을 수 있다.

- http://www.cis.ohio-state.edu/~barr/dnswalk/

```
$ dnswalk -raFl freebsd.org.
Checking freebsd.org.
Getting zone transfer of freebsd.org. from ns2.iafrica.com...done.
SOA=implode.root.com    contact=hostmaster.freebsd.org
FAIL: Cannot get SOA record for pt.freebsd.org from zeus.ci.ua.pt
(lame?): Connection refused
BAD: lv.freebsd.org NS ns.fasts.com: lame NS delegation
BAD: eu.freebsd.org NS server.zsb.th-darmstadt.de: unknown host
WARN: irc3.freebsd.org CNAME irc.nocturnal.net: CNAME (to nocturnal.net)
WARN: ns.th.freebsd.org A 203.154.230.20: no PTR record
WARN: irc2.freebsd.org CNAME va.i-Plus.net: unknown host
1 failures, 3 warnings, 1 errors.
```

Dnswalk는 검사를 수행하기 위해 해당 도메인을 Zone Transfer 한 후 내부적으로 Dig와 Resolver 루틴을 사용한다. Delegation된 도메인을 따라가며 검사를 행하기 때문에 거대 도메인을 관리한다면 한번쯤 사용해봄직 하다.

9Chapter. Miscellaneous

9.1. FQDN(Fully Qualified Domain Name)

FQDN은 명확한 도메인 표기법을 칭한다. 예로 소프트웨어 설치 중 도메인명을 요구하면, YAHOO.COM. 을 입력할지, WWW.YAHOO.COM. 을 입력할지 모호하다. 그래서 이러한 모호성을 피하기 위해 FQDN이란 단어를 사용하며, 이는 Namespace 계층상에서 최종 호스트명을 포함하는 도메인명을 뜻한다.

www(호스트명), yahoo.com.(도메인명), www.yahoo.com.(FQDN)

원칙적으로 도메인의 표기는 네임스페이스상의 경로를 명확히 하기 위해 끝에 도트('.' 루트 도메인)를 포함하여야 하지만, 보통 도트를 생략하고 사용한다.

9.2. DNS와 BIND의 차이

DNS는 Domain Name System의 약자로써, 분산 네이밍 시스템을 뜻한다. 조금 쉽게 풀어보면, 도메인명을 IP 주소로 변환해주는 방법론이다. 즉, 인터넷에 존재하는 수많은 네임서버는 각각 도메인 계층상의 일부분을 관리하고, 정보를 요구하는 클라이언트 Resolver는 규칙에 따라 분산된 자료중 원하는 정보를 찾을 수 있는 시스템, 이 것을 DNS 라고 한다.

BIND는 Berkeley Internet Name Domain의 약자로, DNS를 구현한 소프트웨어의 하나이면서, '워크맨'이란 단어처럼 DNS를 구현한 소프트웨어를 칭하는 대명사로 쓰이기도 한다. BIND는 거의 모든 플랫폼에 포팅되었고, 가장 널리 사용된다.

9.3. BIND-4(Traditional)와 BIND-8(Next Generation) 가지

BIND는 1999년 1월 현재 BIND-4와 BIND-8의 두 가지가 존재한다. 1998년 5월 11일 최종 버전 4.9.7이 릴리즈되며 전통적인 BIND-4 가지는 마감되었고, BIND-8 가지는 현재 8.2를 릴리즈하고 있다. BIND-8은 RFC2136 (ftp://ftp.openbird.com/pub/rfc/rfc2136.txt), RFC1996 (ftp://ftp.openbird.com/pub/rfc/rfc1996.txt)을 수렴하여 메커니즘과 보안이 크게 개선되어 발표되었다. 점진적으로 BIND-8로 옮겨갈 테지만, 현재 대부분의 도메인 메니저와 OS 벤더가 오랜기간 검증된 BIND-4를 선택하고 있으므로, 두 가지는 앞으로도 상당기간 공존할 것으로 예상된다. 하지만, 주 흐름은 BIND-8로 넘어가고 있다.

BIND-4와 BIND-8의 외부적인 차이는, 부트 파일의 변화이다. BIND-4에서는 부트 파일이 named.boot 이고, BIND-8에서는 named.conf 이다. 또한 부트 파일의 작성 방법도 차이가 있다. 기타 Zone 파일과 캐쉬 파일 등은 동일하므로, BIND-4에서 BIND-8로의 마이그레이션은 소프트웨어를 설치하고, 부트 파일을 컨버팅하는 것으로 족하다.

9.4. 퍼블릭 도메인 (Public Domain)

보통 도메인이라 하면 퍼블릭 도메인을 말한다. 이는 인터넷 어디에서나 접속이 가능하도록 네임스페이스 가지 상에 놓여있는 도메인을 뜻한다. 즉, 네임스페이스상에 링크 되지 않은 도메인은 네임서버를 구축하여도 해당 네임서버를 거쳐 직접 resolving하는 경우를 제외하곤 찾을 수 없는 폐쇄 도메인이 된다. 사내에서 보안등의 이유로 간혹 사용된다.

9.5. CNAME의 사용에 관해

CNAME의 사용 여부에 대해 많은 DNS 전문가들은 다른 의견을 보인다. A 레코드만을 사용하자는 의견도 있다. 특히, CNAME이 CNAME을 포인팅하는 경우에 대해서는 관련 자료마다 ’된다/안된다’ 식의 저자 의견이 들어가 있고, 초기 RFC에서는 이를 금하고 있다. 하지만, BIND implementation은 이를 지속적으로 보안 확대하고 있는데 이는 타 도메인의 서비스를 Aliasing하는 경우를 포함하여, CNAME 레코드는 매우 유용한 것이 사실이고 특정 호스트의 IP가 변경되었을 때 관련된 레코드를 찾아다니며 수정을 하기 보다는, CNAME을 사용하여 주 레코드의 수정 내역을 파급시키는 것이 여러모로 편리하고 효율적이기 때문이다. CNAME 레코드의 용법(MX, SOA, NS등의 레코드와 연결하지 않는것)만 준수한다면 CNAME은 매우 유용하게 활용 될 수 있으리라 생각한다.

9.6. Zone 데이터베이스 작성에 대해

일반적으로 다음의 규칙을 준수해 Zone 데이터베이스를 작성하면 실수를 줄이는데 도움이 된다.

- TAB을 사용해 열을 맞춘다. 이것은 빠진 레코드를 찾는데 도움이 된다.
- TTL 값들은 모두 초단위를 사용하거나 2D, 1W와 같이 모두 단위기호를 사용해 일률적으로 기입한다.
- 호스트 정의는 다음과 같이 모두 호스트명만을 사용하거나, 모두 FQDN 표기한다.

```
ftp                     IN      MX      mail
www                     IN      CNAME   power

ftp.nobreak.com.        IN      MX      mail.nobreak.com.
www.nobreak.com.        IN      CNAME   power.nobreak.com.
```

  혹은 좌측은 호스트명을 우측엔 FQDN 표기한다. 타 기관의 호스트를 CNAME으로 연결할 경우가 있기 때문에 이것이 좀더 일반적이고 많이 사용된다.

```
ftp                     IN      MX      mail.nobreak.com.
www                     IN      CNAME   power.nobreak.com.
```

- 가능하면, 알파벳 순서대로 나열하여, 중복 정의되는 부분이 없도록 한다.

9.7. 글루 레코드 (Glue Record)

글루 레코드는 NS 레코드의 인자로 주어지는 A 레코드를 말하며, 네임서버에 부트스트랩 정보를 제공한다. 다음의 경우 ns.nms.nobreak.com 이 글루 레코드이다.

```
nms.nobreak.com.        IN      NS      ns.nms.nobreak.com.
ns.nms.nobreak.com.     IN      A       150.183.110.2       ; 글루 레코드
```

하지만, 다음과 같이 외부에서 관리되는 네임서버 ns.kr.freebsd.org 로 도메인을 위임하는 경우엔, freebsd.org 의 NS에서 글로 레코드 ns.kr.freebsd.org 가 이미 정의되어 있으므로 글루 레코드 ns.nms.nobreak.com 에 ns.kr.freebsd.org 의 IP를 설정하여 부트스트랩 정보로 사용하여서는 안된다.

```
nms.nobreak.com.        IN      NS      ns.nms.nobreak.com. ; (X) 중복된 글
루 레코드
ns.nms.nobreak.com.     IN      A       210.105.79.2

nms.nobreak.com.        IN      NS      ns.kr.freebsd.org.  ; (O) 바른 사용
```

이를 중복된 글루 레코드라 하며, 중복된 글루 레코드는 네임서버가 새로운 IP 주소로 옮겨가거나 없어지는 것을 어렵게 한다. 네임서버에 대한 글루 레코드는 네임스페이스상에 유일하게 유지되는것이 좋다.

9.8. Lame Delegation

Lame delegation이란 Namespace 상에서 깨어진 링크를 말한다.

```
nms.nobreak.com         IN      NS      ns.nms.nobreak.com.
                        IN      NS      ns2.nms.nobreak.com.
```

예를들어 nms.nobreak.com 이 위와 같이 두 개의 네임서버를 갖으나, 두 서버 중 하나 혹은 모두가 해당 도메인에 대한 Authority를 갖지 않는 경우, 즉 Primary, Secondary 설정이 안되어 있을 경우가 Lame delegation에 해당된다.

9.9. Authoritative answer & Non-authoritative answer

Name Server는 질의에 대한 결과를 캐쉬에 저장하고 있기 때문에 같은 질의가 요구되었을 때 Namespace를 뒤지지 않고 캐쉬의 자료로 빠르게 응답한다. 캐쉬의 자료는 Resolving시 얻은 TTL(Time To Live) 시간 동안에만 유효하고, TTL 경과후에는 파기된다. 클라이언트의 도메인 Resolving 요청시 네임서버가 캐쉬의 자료로 응답 할 경우는 Non-authoritative answer이고, 캐쉬에 자료가 없거나, 자료의 TTL이 만기되어 해당 도메인의 Primary 네임서버에서 직접 자료를 얻어 답변을 주었을 경우가 Authoritative answer이다.

9.10. Positive & Negative Caching

실제 생활에서 Resolving 요청은 다음과 같이 많은 부분 중복된다.

```
$ finger @kfug.kr.freebsd.org
$ talk nobreak@kfug.kr.freebsd.org
```

따라서, 네임서버는 한번 검색한 도메인 정보를 캐쉬에 유지하여, 후에 요청될 같은 질의를 효율적으로 대처하도록 구현되어 있다. 그렇다면, 존재하지 않는 도메인에 대한 요청은 어떻게 할까? 일반적으로 잘못된 도메인에 대한 요청도 많이 중복된다. 또한 이 경우 네임서버는 가능한 가지를 모두 탐색하므로, 불필요한 인터넷 트래픽 증가라는 문제도 제기된다. 따라서, 네임서버는 이렇듯 잘못된 쿼리에 대한 결과도 캐싱하여 불필요한 트래픽을 차단한다. 이를 Negative 캐싱이라 하며, 반대로 검색이 되는 도메인에 대한 캐싱을 Positive 캐싱이라 한다.

참고로, 네임서버는 캐쉬를 별도로 저장, 관리하지 않기 때문에 **named**가 종료하면 캐쉬도 함께 사라진다. 따라서, 가능하면 Zone 데이터베이스의 수정후에는 행업(kill -HUP) 시그널을 이용하도록 한다.

9.11. Iterative(Nonrecursive) & Recursive 네임서버

네임서버가 Recursive 모드로 동작할 때에는, 클라이언트(이를 Stub Resolver 라 한다)의 요청에 대해 Namespace를 검색한후 결과를 전달한다. 하지만 Iterative 모드에서는 알 수 없는 질의(자신이 관리하지 않는 도메인에 대한 요청)에 대해, 응답 가능한 NS의 목록을 전달한다. 대부분의 네임서버는 Recursive 모드로 동작하며, Iterative 모드는 루트서버와 같이 네임서버를 위한 네임서버(네임서버간의 통신에는 Iterative 모드가 사용됨)에서 과다한 트래픽을 막기위해 사용한다. 또한, 클라이언트는 Iterative 모드로 설정된 네임서버를 사용할 수 없으므로, 네임서버 목록(예:resolv.conf, 윈도우의 DNS 찾기목록)에 추가하여서는 안 된다. BIND-4에서는 부트파일에 ’options no-recursion’을 추가함으로써, Iterative 모드로 전환할 수 있고, BIND-8의 경우엔 options 엔트리에 ’recursion no;’를 설정한다.

9.12. RTT(Round Trip Time)와 Nameserver 선택

네임서버간에 질의, 응답에 소요되는 시간을 Round Trip Time이라 한다. (Recursive 모드하에서의 총 검색 시간이 아니다) BIND는 내부적으로 타 네임서버에 대한 RTT 값을 기록하고 있다가, 요청 도메인에대한 다수의 Authority NS 중 RTT 값이 가장 낮은 네임서버로 먼저 질의한다. Authority NS들에 대한 RTT 정보를 갖고있지 않을경우엔, 해당 네임서버 전체에 질의(동시에)를 보내어 빠른 응답을 얻음과 함께 부가적으로 RTT를 측정한다. RTT가 측정된 다음부터는 해당 도메인에 대한 요청이 RTT가 가장 적은 서버로 먼저 보내어 진다. 또한, 몇몇 서버만이 계속 사용되는 문제를 막기위해 쿼리를 전송할 때 마다 해당 네임서버에 대한 RTT 값을 조금씩 증가시킨다.

9.13. 와일드카드

참고: RFC1034 (ftp://ftp.openbird.com/pub/rfc/rfc1034.txt) p25

Zone 데이터베이스에는 다음과 같이 와일드카드(*) 사용이 허락된다.

```
*               IN      A       210.105.79.20
```

와일드카드는 Zone에 나타나지 않은 호스트들에 매핑되므로, Zone의 모든 호스트들에 적용되리라 기대하여선 안 된다. 이와 관련된 흔한 실수는 다음과 같은 MX 레코드와의 연결이다.

```
*               IN      MX      mail
```

관리자는 모든 호스트로 배달되는 편지를 한곳으로 모으기 위해, 와일드카드와 MX를 연결하였지만, 이것은 기대한 대로 동작하지 않을 것이다. 기대한 동작을 구현하기 위해서는 모든 호스트에 MX 레코드를 추가하여야 한다. 따라서 본 예는 정의되지 않은 호스트를 목적지로 한 편지를 한곳으로 모을 뿐이다. (때론 유용할 수도 있다)

또한, 와일드카드는 호스트명(도메인 가지의 최 하단)으로만 사용될 수 있다. 다음을 보자.

```
www.*           IN      A       210.105.79.20
```

www.ANYTHING.nobreak.com 의 동작을 기대하였지만, 이 기막힌 아이디어는 불행히도 제대로 동작하지 않는다.

9.14. Serial Number 조정

거대 도메인을 관리하는 메니저들의 실수 중 하나는 잦은 업데이트작업으로 인한 잘못된 Serial 넘버링이다. 일반적인 관례인 YYYYMMDDNN 표기법으로는 4294년까지 표기를 할 수 있는데, 19990205010과 같이 실수로 삽입된 ’0’은 해당 필드를 오버플로우 시킨다. 따라서 Secondary의 Zone은 장기간 업데이트되지 않을 수 있다. 다음과 같이 문제를 해결할 수 있다.

- Secondary를 직접 관리한다면, 먼저 Primary Zone의 Serial을 정상적으로 조정한다. Secondary에 저장되어 있는 Zone 파일(Zone Transfer된)을 삭제한후 BIND를 재 구동한다.
- Secondary가 타기관에 의해 관리되어 앞의 방법이 불가능할 경우, Zone의 Serial을 ’0’으로 설정한다. Secondary는 ’0’을 Serial로 갖는 Zone에 대해서, 무조건적인 업데이트를 강행하므로, Refresh 주기만큼 기다린 후, 다시 정상적인 Serial로 조정하면 된다.

```
nslookup  -type=SOA  kr.freebsd.org  ns2.nobreak.com
```

Secondry가 갖고 있는 해당 Zone의 Serial 번호는 위와 같이 확인할 수 있다.

9.15. IP 변동에 따른 TTL 조정

서비스 되고있는 네트워크에 중요한 변경이 예상된다면, 다음과 같이 해당 호스트의 TTL을 임시로 10분(600sec) 정도로 낮추어 두는 것이 좋다.

```
www     600     IN      A       210.105.79.6
```

타 네임서버가 아예 캐싱하지 않도록 하기 위해 TTL을 0으로 조정하는 것도 나쁘진 않으나, 클라이언트가 해당 도메인을 억세스 할 때마다 반복되는 Resolving을 동반하기 때문에, 바람직한 방법은 아니다. Maximum Propagation Delay Time(조정전 SOA의 Refresh + 조정전의 TTL, 참고: DNS Notify) 만큼 기다린 후, 작업(IP 변경)하면 되는데, 경험적으로 네트워크 변경이 시작되면 예상치 못한 추가 이동이 발생하므로, 네트워크가 안정된 후라도, 1-2일 정도 뒤에 TTL을 원상 복귀하는 것이 좋다. 해당 Zone에 속한 모든 호스트가 대상일 경우엔, SOA 레코드의 Minimum값을 조정하여 일률적으로 적용할 수 있겠다.

10Chapter. Bibliography

1. Paul Albitz & Cricket Liu, "DNS and BIND 2rd Edition", O'Reilly & Associates, Inc., ISBN 1-56592-236-0, Dec 1996 (http://www.ora.com/catalog/dns2/)
2. Paul Albitz & Cricket Liu, "DNS and BIND 3rd Edition", O'Reilly & Associates, Inc., ISBN 1-56592-512-2, Sep 1998 (http://www.ora.com/catalog/dns3/)
3. Paul Albitz, Matt Larson & Cricket Liu, "DNS on Windows NT", O'Reilly & Associates, ISBN 1-56592-511-4, Oct 1998 (http://www.ora.com/catalog/dnswinnt/)
4. Bryan Costales & Eric Allman, "sendmail 2nd Edition", O'Reilly & Associates, ISBN 1-56592-222-0, Jan 1997 (http://www.ora.com/catalog/sendmail2/)
5. Bryan Costales & Eric Allman, "sendmail Desktop Reference", O'Reilly & Associates, ISBN 1-56592-278-6, Mar 1997 (http://www.ora.com/catalog/sendmailqr/)
6. H. Eidnes, G. de Groot & P. Vixie, "Classless IN-ADDR.ARPA delegation", RFC2317, Mar 1998 (ftp://ftp.openbird.com/pub/rfc/rfc2317.txt)
7. M. Andrews, "Negative Caching of DNS Queries (DNS NCACHE)", RFC2308, Mar 1998 (ftp://ftp.openbird.com/pub/rfc/rfc2308.txt)
8. O. Vaughan, "A Legal Basis for Domain Name Allocation", RFC2240, Nov 1997 (ftp://ftp.openbird.com/pub/rfc/rfc2240.txt)
9. M. Hamilton, "Use of DNS Aliases for Network Services", RFC2219, Oct 1997 (ftp://ftp.openbird.com/pub/rfc/rfc2219.txt)
10. R. Elz, R. Bush & S. Bradner, "Selection and Operation of Secondary DNS Servers", RFC2182, Jul 1997 (ftp://ftp.openbird.com/pub/rfc/rfc2182.txt)
11. S. Thomson, Y. Rekhter & J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC2136, Apr 1997 (ftp://ftp.openbird.com/pub/rfc/rfc2136.txt)
12. P. Vixie, "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC1996, Aug 1996 (ftp://ftp.openbird.com/pub/rfc/rfc1996.txt)
13. C. Everhart, L. Mamakos & R. Ullmann, "New DNS RR Definitions", RFC1183, Oct 1990 (ftp://ftp.openbird.com/pub/rfc/rfc1183.txt)
14. P. Mockapetris, "Domain Names - Implementation and Specification", RFC1035, Nov 1987 (ftp://ftp.openbird.com/pub/rfc/rfc1035.txt)
15. P. Mockapetris, "Domain Names - Concepts and Facilities", RFC1034, Nov 1987 (ftp://ftp.openbird.com/pub/rfc/rfc1034.txt)
16. M. Lottor, "Domain Administrators Operations Guide", RFC1033, Nov 1987 (ftp://ftp.openbird.com/pub/rfc/rfc1033.txt)
17. M. Stahl, "Domain Administrators Guide", RFC1032, Nov 1987 (ftp://ftp.openbird.com/pub/rfc/rfc1032.txt)