안전한 DNS에서의 효율적인 동적 갱신과 존 전송 기능의 설계와 구현

심 희 원[†] · 심 영 철[††] · 임 찬 순[†††] · 이 만 희[†††] · 변 옥 환[††††]

요 약

안전한 DNS(Domain Name System)는 기존 DNS에 비해 관리해야 할 존(Zone)의 정보가 크게 늘었고 이들 사이의 종속 관계도 매우 복잡하게 되어, 효율적으로 존 정보를 관리할 수 있는 메커니즘이 필요하게 되었다. 또한 안전한 DNS를 다른 인터넷 응용 서비스와 효율적으로 접속할 수 있도록 하려면 일관성 있는 인터페이스가 필요하다. 이에 따라 본 논문에서는 DNS의 존 정보를 동적으로 추가하고 삭제하는 동적 갱신과, 갱신된 존 정보를 DNS 서버간에 효율적으로 전송할 수 있는 존 전송의 기능을 추가한 확장된 안전한 DNS를 설계하고 구현하였다. 그리고 효율적인 존 전송을 위해 전체 존 전송과 점진적 존 전송의 두 가지 존 전송을 통합하는 방법과 함께 존 전송 메시지의 압축 방법을 제시하였고, 존 전송과 동적 갱신을 동시에 제공하기 위해 델타파일이라는 새로운 자료구조를 제시하였다.

Design and Implementation of Efficient Dynamic Update and Zone Transfer in the Secure DNS

Hee-Won Shim[†] · Young-Chul Shim[††] · Chan-Soon Im[†††] · Man-Hee Lee[†††]
Ok-Hwan Byeon[††††]

ABSTRACT

In the secure DNS the amount of information that should be managed greatly increased and the interdependency in the information became very complex. Therefore, it became necessary to develop a mechanism which can manage zone information efficiently. Moreover, a consistent interface became also necessary so that a secure DNS may be efficiently interconnected with other Internet application services. In this paper we explain the design and implementation of a secure DNS extended with two functions : (1) a dynamic update function which enables to add and remove zone information dynamically and (2) a zone transfer function that efficiently transfers updated zone information among DNS servers. We developed a method which integrates two zone transfer mechanisms, full zone transfer and incremental zone transfer, and also proposed a method to compress data in the zone transfer message. We also introduced a data structure called a delta file to integrate the zone transfer function and the dynamic update function.

1. 서 론

인터넷은 이미 모든 국가들의 기반구조가 되어 그 활용도는 과거의 단순한 홍보나 정보의 공유에서 현재에는 전자상거래까지 활발히 진행되고 있다. 따라서 해커들의 침입에 의해 시스템이 파괴되거나 중요한 정보가 유출되는 등 심각한 보안사고도 많이 발생하고 있다. 이러한 보안사고를 막기 위한 방법은 여러 부분에서 활발히 진행되고 있지만, 인터넷을 사용할 때의

† 준 회 원 : 홍익대학교 대학원 전자계산학과
†† 종신회원 : 홍익대학교 정보컴퓨터공학부 교수
††† 정 회 원 : 연구개발정보센터 슈퍼컴퓨팅사업단
†††† 종신회원 : 연구개발정보센터 슈퍼컴퓨팅사업단
논문접수 : 1999년 9월 11일, 심사완료 : 1999년 12월 9일

시작부분이라 할 수 있는 DNS(Domain Name System)에서의 보안이 가장 시급한 분야 중의 하나라고 할 수 있다.

DNS는 도메인 이름을 IP(Internet Protocol) 주소로 또는 그 역으로 변환시켜주는 개방형 분산시스템 형태를 지닌 인터넷 응용 계층에서의 서비스이다. 따라서 DNS는 인터넷에서의 가장 중요한 기반구조라 할 수 있지만 보안을 제공하기 위한 연구는 근래에 들어서야 시작된 상황이다.

IETF(Internet Engineering Task Force)는 안전한 DNS 서비스를 위하여 기존 DNS의 네이밍 서비스 기능 외에 다음의 세 가지 보안 기능이 추가될 것을 권고하고 있다. 첫째는 인증된 공개키의 분배 서비스이고, 두 번째는 서버에 저장된 내용의 무결성과 데이터 인증 서비스이며, 마지막은 질의 또는 응답 메시지의 무결성을 제공하는 것이다[8]. 이러한 보안 서비스의 추가로 안전한 DNS는 인터넷 보안에 크게 기여하게 되었고, 기존의 네이밍 서비스 이외에 PKI(Public Key Infrastructure) 등 그 활용범위가 매우 넓어지게 되었다[10].

이와 같이 안전한 DNS는 기존의 네이밍 서비스에 보안 서비스를 제공하게 되어 큰 의미를 지니지만, 반면에 이에 따른 문제점도 많이 발생하게 된다. 이들 중 가장 큰 문제점은 상당히 부피가 커진 존(Zone) 파일과 이 존에 속한 여러 자원 레코드들 사이의 종속관계가 매우 복잡해져서 존 파일을 갱신하기가 어려워진 것이다. 따라서 동적 갱신(Dynamic Update)[6]에 의해 자동적으로 존 정보를 추가하고 삭제하는 기능이 필요하게 되었다.

이러한 동적 갱신은 효율적이고 안전하게 존을 관리하게 할 뿐 아니라 다양한 인터넷 응용 프로그램과 접속하여 DNS의 기능을 확장시킬 수 있도록 할 수 있다. 즉, DNS를 DHCP(Dynamic Host Configuration Protocol) 서버나 인증서 서버(Certificate Authority Server), 멀티캐스트 에이전트(Multicast Agent) 등의 다른 인터넷 서비스와 효율적으로 접속하여 보다 향상된 서비스를 제공할 수 있다[11]. 이렇게 안전한 DNS가 다른 여러 인터넷 응용 서비스와 연동하여 사용되려면 DNS 데이터베이스에 대한 신뢰성 있는 관리와 보안 서비스를 제공하는 것 이외에 자주 변동되는 존의 내용을 일관성 있고 효율적으로 관리하는 기능이 필수적인 요소가 된다. 즉, 동적 갱신이 제공되지 않는 DNS를 수동으로 관리하려면 관리자의 많은 시간과 노력이 필요할 뿐만 아니라 판단 착오에 의한 보안 사고도 클 것이다.

동적 갱신에 의해 변경된 DNS 데이터베이스는 존 정보의 일관성을 보장하기 위해 보조 DNS 서버에 가능한 한 빨리 변경된 내용이 전달되어야 한다. 따라서 일정 시간 후에 주기적으로 존의 변경 내용을 자동적으로 보조 DNS 서버에 전파하는 존 전송(Zone Transfer)의 방식이 제안되었다. 하지만 기존의 존 전송방식이 변경된 내용의 크기에 상관없이 존의 내용을 모두 전송하는 전체 존 전송(Full Zone Transfer) 방식만을 제공하여 존의 작은 변화를 전파하는 데는 효율적이지 못하였다[4]. 따라서 존의 변경된 부분만을 추출하여 전송하는 점진적 존 전송(Incremental Zone Transfer) 방식이 제안되었다. 이러한 동적 갱신과 존 전송은 안전한 DNS내에서 구현되기 위해 보안 기능이 확장된 안전한 동적 갱신과 존 전송이 되어야 한다.

본 구현을 위해서는 안전한 DNS의 개발이 선행되어야 한다. 따라서 국내 기술로 개발된 DNS에 암호화 기능을 추가하여 IETF에서 요구하는 세 가지의 보안 기능을 제공하는 안전한 DNS를 이미 개발하였다[9].

본 논문에서는 이러한 안전한 DNS를 기반으로 확장되는 기능인 보안성이 제공되는 동적 갱신과 존 전송 기능의 설계 및 구현에 대해 설명한다. 이 두 기능의 추가를 위해서 먼저 확장된 안전한 DNS의 구조를 정의하였고, 알고리즘을 개발하였으며, 라이브러리 함수를 정의 및 구현하였다. 그리고 동적 갱신과 존 전송을 효율적으로 제공하기 위해 델타파일(Delta File)이라는 새로운 자료구조를 제시하였고, 존 전송 메시지의 압축 방법을 소개하였으며, 기존의 존 전송 방식과 호환성을 유지하면서도 두 가지 존 전송방식을 동시에 운용하기 위한 효율적인 존 전송 메커니즘에 대한 방안을 제시하였다. 따라서 완성된 안전한 DNS는 보안 서비스를 제공하는 강력한 동적 갱신으로 효율적인 존 관리 메커니즘을 제공하고, 안전한 존 전송에 의해 신뢰성 있고 효율성이 높은 존의 전송을 보장할 수 있도록 하였다. 이와 같이 확장된 안전한 DNS는 실제로 구현되었고 그 활용 방법과 실험 결과에 대해 설명하였다.

본 논문의 구성은 다음과 같다. 제2장에서는 IETF 문서를 기반으로 동적 갱신과 존 전송에 대한 개략적인 설명을 하고 이 두 가지 기능이 확장된 안전한

DNS의 전체적인 구조를 설명한다. 제3장에서는 안전한 동적 갱신에 대한 설계 및 구현에 대해 기술하고, 제4장에서는 안전한 존 전송에 대한 설계 및 구현을 설명한다. 제5장에서는 구현된 기능을 활용하는 방안과 구현된 시스템의 실험 결과에 대해 설명하며, 마지막 제6장의 결론으로 본 논문의 끝을 맺는다.

2. 확장된 안전한 DNS의 기능과 구조 및 응용

본 장에서는 IETF의 문서를 기반으로 본 논문에서 가장 기초가 되는 안전한 DNS에 대해 간단히 설명하고 다음으로 확장되는 기능인 동적 갱신과 존 전송에 대해 설명한다. 그리고 이 두 기능이 추가된 DNS의 전체적인 구조에 대해 소개한다.

2.1 안전한 DNS

RFC2565에 정의된 안전한 DNS는 다음 세 가지의 보안 서비스를 제공한다.

- 키 분배 : 존(Zone), 호스트, 사용자들은 한 쌍의 공개키(Public Key)와 개인키(Private Key)를 보유하고 이 중 공개키는 DNS 서버에 저장되며 사용자의 요구에 따라 분배된다. 이러한 공개키나 개인키는 KEY 자원레코드[3]로 DNS 데이터베이스에 저장된다.
- DNS 서버에 저장된 내용에 대한 무결성과 데이터 인증 : DNS 서버 내에 저장되어 있는 자원 레코드들은 DNS 서버에 의해 전자서명되고 이 전자서명은 별도의 SIG(SIGnature) 자원 레코드로서 저장된다. 또한 NXT(NeXT) 자원레코드는 해당 자원레코드가 없다는 것을 인증하는 용도로 사용된다.
- DNS 질의와 응답에 대한 무결성과 데이터 인증 : DNS 질의와 응답 패킷의 헤더와 내용은 DNS 클라이언트와 서버에 의해 서명되어 전달된다. 따라서 패킷의 전자서명을 검증하여 패킷의 진위 여부를 알 수 있게 된다.

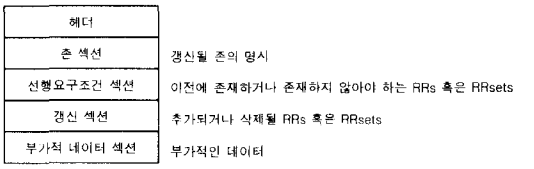
즉, 안전한 DNS는 기존 DNS가 제공하던 자원레코드와 함께 KEY, SIG, NXT 자원레코드가 함께 전송되어 앞에서 언급한 3가지 보안 서비스를 제공하게 되는 것이다[8].

2.2 동적 갱신

DNS는 본래 정적으로 설정된 데이터베이스에 대한 질의를 지원하기 위해 설계되었다[1, 2]. 따라서 데이터베이스의 모든 갱신은 일률적으로 관리자에 의해 존의 마스터 파일을 수동적으로 편집하여 이루어졌다. 하지만 DNS 서버가 DHCP 서버, 또는 인증서 서버 등과 연동되려면 DNS의 존 파일은 매우 빈번하게 변경될 수밖에 없다. 그러므로 이러한 갱신을 담당하는 서버에 의해 자동적으로 갱신한다면 DNS의 활용도가 더욱 높아질 것이다.

동적 갱신 기능은 이름이 명시된 존의 자원레코드들(이후, RRs : Resource Records) 혹은 같은 이름을 가지는 자원레코드의 집합(이후, RRsets)을 추가하거나 삭제하는 것을 동적으로 가능하게 한다. 하지만 이러한 갱신이 권한이 없는 주체에 의해서 이루어지거나, 또는 DNS 데이터베이스의 내용과 상반되는 갱신을 허용해서는 안될 것이다. 따라서 갱신 이전에 권한 검사나 오류를 일으킬 수 있는 요청을 판단하여 내부적인 정책에 의해 갱신 여부를 판단하여야 한다.

동적 갱신 메시지 형식은 (그림 1)과 같이 표준 DNS 메시지를 일부 변형하여 사용하고 있다.



(그림 1) 동적 갱신 메시지 형식

- 헤더 : 헤더 부분은 DNS 메시지의 헤더 구조와 동일하다[5]. 변경된 내용은 동작번호(OPCODE)가 갱신(UPDATE)으로 바뀌고, 응답코드(RCODE)는 동적 갱신의 오류를 나타내기 위해 5가지가 추가되었다.
- 존 섹션(Zone Section) : 이 메시지에 의해 갱신될 존을 명시한다.
- 선행요구조건 섹션(Prerequisite Section) : 갱신을 하기 위해서 요구되어지는 선행 조건들을 명시한다. 즉, 자원레코드를 삭제하거나 추가하기 위해 미리 그 존에 해당 자원레코드가 있는지 여부를 알아 중복된 자원레코드를 추가하거나, 존재하지 않는 자원레코드를 삭제하는 오류를 범하지 않도록 한다.
- 갱신 섹션(Update Section) : 실제 갱신이 이루어질 정보를 기술하고 있다. 이 부분에 속하는 자원레코드들은 DNS에 의해 실제 내부 데이터베이스에 추

가되거나 삭제된다.

- 부가적 데이터 섹션(Additional Data Section) : 동적 갱신 이외에 필요한 자원레코드를 추가할 수 있도록 하는 부분이다. 보통 사용되지 않는 부분이지만, 보안 서비스를 제공하기 위해 메시지를 전자 서명하여 자원레코드 형태로 부가적 데이터 섹션에 추가한다.

동적 갱신이 안전한 DNS에서 보안서비스를 제공하기 위해 확장되어야 할 사항은 크게 2가지로 구분되어질 수 있다. 첫째는 갱신을 요청한 주체에 대한 인증 서비스이다. 즉, 갱신을 요청할 주체는 미리 DNS 데이터베이스에 자신의 공개키를 저장하여야 한다. 이 공개키에는 서명필드(Signatory Field)라는 특별한 4비트의 필드가 있어, 해당 주체가 그 존을 갱신할 수 있는지 여부뿐만 아니라 갱신의 범위까지 알 수 있게 된다.

둘째로는 갱신 요청메시지의 진위를 확인할 수 있는 전자 서명을 제공하는 것이다. 즉, 갱신 요청이나 응답 메시지에는 모두 전자서명이 포함되어 메시지에 대한 진위 여부를 판단할 수 있도록 하며, 갱신이 요청되는 모든 자원레코드에 대해서도 모두 전자서명을 하여 침입자에 의해 내부 데이터베이스 내용이 변조된 사실이 없음을 인증할 수 있도록 한다.

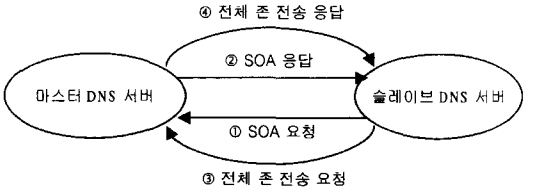
2.3 존 전송

존 전송이란 마스터(Master) DNS 서버의 내용 변화를 슬레이브(Slave) DNS 서버에게 전달하여 내용의 일관성을 보장하도록 하는 기능이다. 즉, 슬레이브 DNS 서버는 주기적으로 마스터 DNS 서버의 존 정보가 변경되었는지를 확인하고, 만약 변경되었을 경우 마스터 DNS 서버에게 변경된 내용을 요청하여 내부 데이터베이스를 갱신하도록 한다.

이러한 존 전송의 방식은 크게 두 가지로 나뉠 수 있다. 변경된 부분과는 상관없이 전체 존의 정보를 모두 전송하는 전체 존 전송(Full Zone Transfer)과 변경된 부분만을 추출하여 보다 효율적으로 존을 전송할 수 있도록 하는 점진적 존 전송(Incremental Zone Transfer)이 그것이다.

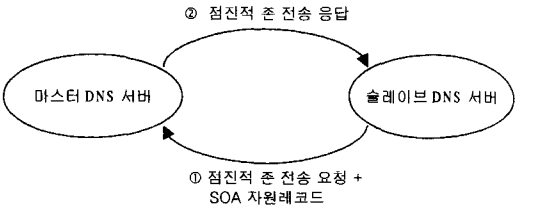
점진적 존 전송 방식은 전체 존 전송에 비해 구현이 어렵고 갱신된 존의 내용이 많을 때 비효율적이지만 실제 상황에서는 변경되는 내용이 많지 않으므로 대부분의 경우 점진적 존 전송방식이 보다 효율적인 방법이라 할 수 있다.

전체 존 전송은 세부적으로 (그림 2)와 같이 동작한다. 슬레이브 DNS 서버는 해당 존을 담당하는 SOA (Start Of Authority) 자원레코드의 갱신(Refresh)값을 초단위로 계산하여 그 값이 0으로 된 시점에서 마스터 DNS 서버에게 SOA 자원레코드를 요청하게 된다. 슬레이브 DNS 서버는 응답으로 받은 SOA 자원레코드의 일련번호(Serial) 값이 자신의 SOA 자원레코드의 일련번호보다 크다면 마스터 DNS 서버에 변경된 내용이 있다는 것을 의미하는 것이므로 존 전송을 요청하게 된다. 마스터 DNS 서버는 내부 데이터베이스를 모두 검색하여 모든 자원레코드를 하나씩 전송하게 된다. 마지막으로 슬레이브 DNS 서버는 전송 받은 존 정보를 존 파일에 저장하고 DNS를 재구동시켜 변경된 내용을 내부 데이터베이스에 반영시킨다.



(그림 2) 전체 존 전송방식

점진적 존 전송방식은 이와 달리 (그림 3)과 같이 보다 함축된 방식으로 동작하게 된다. 슬레이브 DNS 서버는 해당 존을 담당하는 SOA 자원레코드의 TTL (Time-to-live)이 기간 만료되면 자신의 존에 대한 갱신이 필요하다고 간주하고, 마스터 DNS 서버에게 점진적 존 전송을 요청한다. 이때 슬레이브 DNS 서버는 존 전송 요청과 함께 자신이 관리하는 존의 SOA를 전송한다. 따라서 마스터 DNS 서버는 슬레이브 DNS의 SOA 자원레코드에 포함된 일련번호를 자신의 일련번호와 비교하여 존 전송이 필요한지 여부를 판단할 수



(그림 3) 점진적 존 전송방식

있게 된다. 이후에 마스터 DNS 서버는 변경된 내용만을 전송하게 된다.

존 전송에 보안서비스를 확장하는 방법 역시 기본적인 안전한 DNS의 질의/응답과 크게 다를 것이 없다. 즉 질의 또는 응답 메시지에 전자 서명을 포함하여 내부 데이터베이스의 무결성과 메시지의 인증 서비스를 제공할 수 있게 된다. 하지만 점진적 존 전송에서 인증 서비스를 제공하려면 보다 복잡한 과정을 거쳐야 한다. 그 이유는 자원레코드의 부재를 인증할 수 있는 NXT 자원레코드 때문이다[8]. 기본적으로 NXT 자원레코드는 다음과 같이 사용된다.

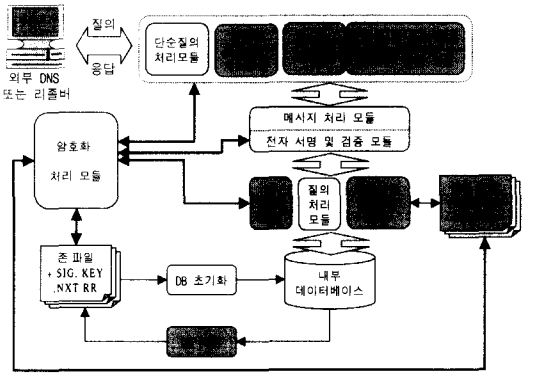
a.zone NXT c.zone
a.zone SIG {NXT 자원레코드를 전자서명}

즉, a.zone과 c.zone에는 어떠한 자원레코드도 없음을 NXT와 SIG 자원레코드의 쌍으로 인증하고 있다. 따라서 b.zone을 질의하게 된다면 위의 두 자원레코드가 응답되어 안전하게 b.zone 자원레코드가 없음을 인증할 수 있다. 따라서 점진적 존 전송에 의해 존이 갱신된 다면 기존의 데이터베이스의 변경으로 NXT 자원레코드의 체인이 손상되므로 NXT와 SIG의 체인을 재구성하는 별도의 작업이 필요하게 된다.

2.4 확장된 안전한 DNS의 구조

동적 갱신과 점진적 존 전송 기능을 수행하는 확장된 DNS의 전체적인 구조는 (그림 4)와 같다. 구현된 DNS는 보안 서비스를 제공하므로 외부 DNS 또는 리졸버로부터의 질의와 응답에 대한 메시지 인증 서비스를 제공받는다. 또한 내부 데이터베이스의 모든 내용에 대한 전자서명이 수행되므로 안전한 네이밍 서비스를 제공하는 기반이 된다.

본 구현에서는 안전한 DNS를 기반으로 보안성이 제공되는 동적 갱신 모듈과 존 전송 모듈을 확장시켜 효율적인 서비스를 제공하도록 하였다. 또한 동적 갱신 메시지에 의해 내부 데이터베이스의 정보를 직접 추가 또는 삭제할 수 있도록 DB 갱신 모듈을 추가하였고, 동적 갱신에 의한 정보를 다른 DNS에 전파하도록 델타파일 입출력 모듈을 추가하였다. 존 전송 모듈은 두 가지 존 전송 방식을 효율적으로 운용하도록 존 전송 선택 모듈이 확장되었고, 보안성을 제공하기 위해 암호화 처리 모듈과 통신하여 모든 메시지에 전자서명을 수행하도록 하였다. (그림 4)의 음영 처리된 부분은 동적 갱신과 존 전송을 위해 본 구현에서 새롭게 추가된 부분으로 이 모듈들은 다음 장에서 자세히 설명하도록 한다.



(그림 4) 확장된 안전한 DNS의 구조

3. 동적 갱신의 설계와 구현

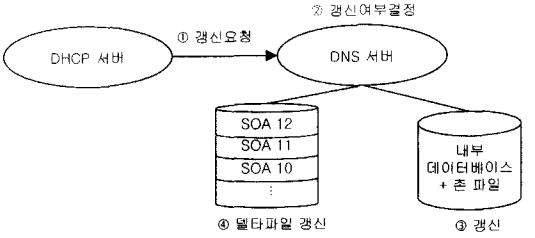
본 장에서는 동적 갱신을 위해 필요한 내부 데이터베이스의 변경 모듈과 동적 갱신 요청을 처리하는 모듈의 설계 및 구현에 대해 자세하게 설명한다.

3.1 동적 갱신의 설계

동적 갱신은 (그림 5)와 같이 4단계에 걸쳐 이루어진다. 가장 먼저 DHCP 서버와 같은 외부 서버에 의해 동적 갱신이 요청된다. 이러한 요청 메시지에는 갱신을 요청하는 주체에 대한 정보와 이를 인증하는 내용이 들어있고, 또한 선행 요구조건과 보안 서비스를 제공하기 위한 전자서명들이 포함되어 있다. 따라서 갱신 요청을 받은 DNS 서버는 갱신을 수행하기 이전에 이러한 갱신 요청이 정당한 것인지 판단하여야 한다.

DNS 서버가 동적 갱신의 여부를 결정하기 위한 조건에는 3가지가 있다. 첫 번째는 그 요청이 보안성에 문제가 없는 요청인지 판단하여야 한다. 따라서 메시지에 포함된 SIG 자원레코드를 일일히 검증하는 작업이 선행되어야 한다. SIG 자원레코드의 전자서명이 모두 검증된 후에는 이러한 요청을 하는 주체의 갱신키(Update Key)를 가지고 이러한 요청이 정당한지의 여부를 판단한다. 즉, 갱신키에 포함된 서명필드(Signatory Field)를 보고 이러한 주체가 어느 정도의 권한을 가지고 있는지 파악하여야 한다. 만약, 해당 이름의 자원레

코드만을 변경할 수 있는 권한을 가진 주체가 전체 존에 대한 갱신을 요청했을 경우 무시되어져야 한다. 마지막 검사는 동적 갱신 메시지에 포함된 선행요구섹션(Prerequisite Section)의 내용에 따라 내부 데이터베이스에서 해당 자원레코드의 존재 여부를 파악하여야 한다.



(그림 5) 동적 갱신 작업의 순서

이러한 3가지 검사가 성공적으로 끝난 후에야 갱신 섹션(Update Section)의 내용을 내부 데이터베이스에 갱신하게 된다. 만약 오류가 발생하였을 경우 요청 메시지의 응답코드에 해당 오류코드를 삽입하여 응답한다. 이때 주의하여야 할 사항은 동적 갱신에 의해 내부 데이터베이스의 내용이 변경되었으므로 존의 SOA 자원레코드의 일련번호를 자동적으로 증가시켜야 한다는 점이다. 만약 일련번호가 증가되지 않는다면 슬레이브 DNS 서버는 동적 갱신에 의해 존이 변경된 사항을 알 수 없게 된다. DNS 서버의 신뢰성을 보장하기 위해서 내부 데이터베이스에 동적으로 변경된 사항은 시스템의 재구동 이후에도 갱신된 내용을 가져오도록 존 파일에도 저장한다.

보통의 동적 갱신은 세 번째 갱신 단계에서 모든 작업이 끝나게 된다. 하지만 본 논문에서는 동적 갱신과 존 전송을 동시에 고려하기 때문에 마지막으로 델타파일이라는 새로운 자료 구조에 변경된 내용을 저장하는 작업이 필요하게 된다. 동적 갱신에 의해 갱신된 내용은 새로 일련번호가 증가되어 만들어진 SOA 자원레코드와 함께 순차적으로 델타파일에 저장되게 된다. 이러한 델타파일에는 변경된 존 정보에 해당되는 SIG와 NXT 자원레코드가 포함되어 완전한 형태로 저장된다.

안전한 DNS에서는 존의 정보를 저장하기 위해 크게 3가지의 자료구조를 가지게 된다. 첫 번째 자료구조는 가장 전통적인 자료구조로서 안전한 DNS가 처음 구동시 필요한 파일형태의 존 파일(Zone File 또는 Boot File)이 있으며, 두 번째로는 이러한 존 파일에서 존의 정보를 읽어와 직접 접근하여 처리가 가능한 연결리스트(Linked List) 형태로 구성된 자료구조인 내부 데이터베이스가 있다. 앞으로 언급되는 내부 데이터베이스란 이러한 연결리스트 형태의 자료구조를 지칭한다. 마지막으로 점진적 존 전송 방식을 위해 고안된 델타 파일(Delta File)은 동적 갱신에 의해 변경된 부분만을 저장하는 자료구조이다.

안전한 DNS는 존의 부피가 상당히 커졌기 때문에 효율적인 존의 관리가 필수적이라 할 수 있다. 따라서 이러한 존 정보를 내부 데이터베이스 형태로 변환하기 위해 도입된 자료구조는 메모리 사용을 최소한으로 줄일 수 있는 연결리스트를 사용하였다. 또한 존 정보에 대한 접근이 빈번하므로, 신속한 접근을 위해 해쉬 기법을 사용한다. RNAME과 RDATA는 내부 데이터베이스를 구성하는 가장 중요한 자료구조이다. RNAME은 자원레코드의 이름을 저장하고 있으며 같은 이름의 자원레코드가 여러 개 있을 수 있으므로 하나의 RNAME은 자원레코드의 내용을 저장하는 RDATA의 포인터를 여러 개 가지고 있다.

델타 파일은 내부 데이터베이스의 변경내용을 차례대로 저장하여 점진적 존 전송을 하기 위한 파일이다. 따라서 존 파일과는 달리 내부적으로 처리되므로 텍스트 형식으로 저장될 필요가 없다. 이러한 이유로 델타파일은 DNS 메시지의 형태로 이진 파일(Binary File)에 저장되고, 필요시 직접 메시지로 활용할 수 있다. 자세한 내용은 이후에 존 전송 모듈의 구현과 설계에서 설명하겠다.

동적 갱신이나 존 전송에 의해 내부 데이터 베이스가 변경되어야 할 경우 RNAME과 RDATA를 추가하거나 삭제하며, SIG와 NXT 자원레코드의 체인을 재구성하는 기능을 수행하는 모듈을 내부 데이터베이스 변경 모듈이라 지칭한다.

3.2 동적 갱신 모듈의 구현

동적 갱신 모듈은 메시지의 각 섹션을 처리하는 동적 갱신 처리 모듈과 갱신 섹션의 자원레코드를 실제 내부 데이터베이스에 추가하거나 삭제하는 내부 데이터베이스 변경 모듈로 나눌 수 있다. 이 두 가지 모듈이 구현된 내용을 의사코드(Pseudo Code) 형태로 설명하도록 하겠다.

3.2.1 동적 갱신 처리 모듈의 구현

동적 갱신 처리 모듈은 존 섹션의 처리와 선행 요구

조건 섹션의 처리, 그리고 요청자의 권한 검사로 크게 3부분으로 나뉜다.

(그림 6)은 존 섹션에 대한 처리 과정을 나타내고 있다. 즉, 동적 갱신은 존 정보에 대한 일관성을 위해 마스터에서만 발생할 수 있으므로 슬레이브일 경우에는 마스터로 메시지를 전달하고, 마스터일 경우 존의 갱신 작업을 계속 진행하도록 한다.

```
if ( 존 섹션의 자원레코드종류 != SOA) return (형식 오류)
    if ( 해당 존의 종류 == SLAVE) return forward()
    if ( 해당 존의 종류 == MASTER) return update()
    return (권한 오류)
```

(그림 6) 존 섹션에 대한 처리

만약, 특정 자원레코드를 삭제하려 할 때 삭제할 자원레코드가 내부 데이터베이스에 없다면 존재하지 않는 자원레코드를 삭제하려는 오류가 발생하게 된다. 따라서 선행 요구조건 섹션에서는 내부 데이터베이스에 존재 여부를 미리 파악하여 모든 요구조건이 만족한 후에야 비로소 본격적인 동적 갱신이 허용하도록 한다. 만약 조건이 만족하지 않는다면 해당 응답코드를 요청자에게 돌려보내야 한다. 본 구현에서는 선행 요구조건 섹션에 있는 모든 자원레코드들에 대해 클래스(Class)와 종류(Type) 필드를 검사하고 이들 각각에 대해 내부 데이터베이스의 자원레코드와 대조하여 그 존재 유무를 판단하도록 하였다.

요청자의 권한 검사는 존 키와 갱신키의 서명 필드를 확인하여 처리한다. 즉, 존 키가 동적 갱신을 허용하는지 여부에 대한 사항을 확인하고 동적 갱신의 방법을 선택하게 된다. 또한 동적 갱신을 요청한 주체에 대한 공개키를 내부 DNS 데이터베이스에서 찾아 해당 키의 서명 필드를 체크하여 권한 밖의 요청을 하는지 검사하여야 한다.

3.2.2 내부 데이터베이스 변경 모듈의 구현

내부 데이터베이스 변경 모듈은 동적 갱신 메시지의 갱신 섹션에서 자원레코드를 하나씩 추출하여 내부 데이터 베이스에 추가하거나 삭제하도록 한다. 자원레코드의 실제적인 추가와 삭제는 ldb_update라는 함수를 호출하여 수행한다.

```
ldb_update (Rdata *rd, u_char *name, int op)
```

ldb_update()는 내부 데이터베이스에 RDATA 구조체를 직접 추가하거나 삭제하는 기능을 하고 마지막으로 존 파일을 현재의 내부 데이터베이스의 내용으로 갱신하여야 한다. 따라서 실제 추가되거나 삭제될

```
for 업데이트 섹션에 존재하는 모든 자원레코드들에 대하여 loop
    if (자원레코드의 class 필드 == 존의 SOA 자원레코드의 class 필드)
        if (자원레코드의 종류 필드 == CNAME)
            if (해당 이름의 CNAME 자원레코드가 내부 데이터베이스에 부재) next 자원레코드
        elsif (해당 이름의 CNAME 자원레코드가 내부 데이터베이스에 존재) next 자원레코드
        if (자원레코드의 종류 필드 == SOA)
            if (해당 이름의 SOA 자원레코드가 내부 데이터베이스에 부재 ||
                내부 데이터베이스의 SOA 자원레코드 일련번호 필드 >  자원레코드의 일련번호 필드)
                next 자원레코드
        sdb_update(자원레코드, DYN_ADD_RR)
    elsif (자원레코드의 class 필드 == ANY)
        if (자원레코드의 type 필드 == ANY)
            if (자원레코드의 이름 == 존의 이름) sdb_update(자원레코드, DYN_DEL_ALLRR)
        elsif (자원레코드의 이름 == 존의 이름 &&
            (존의 종류 필드 == SOA || NS)) next 자원레코드
        else sdb_update(자원레코드, DYN_DEL_RR)
    elsif (자원레코드의 class 필드 == NONE)
        if (자원레코드의 type 필드 == SOA) next 자원레코드
        if (자원레코드의 type 필드 == NS &&
            해당 이름의 NS 자원레코드가 내부 데이터베이스에 존재) next 자원레코드
        sdb_update(자원레코드, DYN_DEL_RRSET)
return (오류 없음)
```

(그림 7) 갱신 섹션에 대한 처리

RDATA와 자원레코드의 이름, 마지막으로는 실행코드를 입력변수로 갱신을 수행한다. 실행 코드는 다음과 같이 4가지로 정의되어 있다[4].

- DYN_ADD_RR : 하나의 자원레코드를 내부 데이터베이스에 추가한다.
- DYN_DEL_RRSET : 해당 종류의 자원레코드 집합을 모두 내부 데이터베이스에서 삭제한다.
- DYN_DEL_RR : 해당 종류와 일치하는 RDATA를 내부 데이터베이스에서 삭제한다.
- DYN_DEL_ALLRR : 해당 이름을 가지는 모든 자원레코드를 내부 데이터베이스에서 삭제한다.

(그림 7)은 갱신 섹션에 대한 처리를 나타낸다. SOA, CNAME(Canonical NAME) 등의 자원레코드는 종속성과 보안성의 문제점들 때문에 갱신을 허용하지 않는다. 실제적인 갱신은 sdb_update()에 의해 수행되며 갱신이전에 SIG와 NXT자원레코드를 생성시키는 작업을 수행한다. 따라서 동적 갱신이 수행된 이후에도 존의 정보는 무결성과 인증 서비스를 안전하게 제공할 수 있게 된다.

3.2.3 구성 함수별 세부 구현사항

동적 갱신 모듈을 구성하는 함수들은 아래와 같다. 즉, 동적 갱신 메시지를 분석하여 각 섹션별로 처리하는 동적 갱신의 API 함수와 내부 데이터베이스에 해당 자원레코드의 존재 유무를 판단하는 함수들이 제공된다. 또한 동적 갱신 메시지를 내부 데이터베이스에 실제적으로 갱신을 수행하는 내부 데이터베이스 갱신 함수와 그밖에 이를 보조하는 여러 함수들로 구성되어 있다.

- dyn_update() : 동적 갱신 요청 메시지를 전반적으로 처리하는 API 함수이다.
- zone_type() : 존 섹션을 처리하기 위한 함수로 해당 존의 종류를 알아낸다.
- find_zone_rr() : 자원레코드가 내부 DB에 있는지 여부를 알아낸다.
- check_update_permission() : 동적 갱신을 요청하는 개체의 공개키를 검사하여 권한을 가지고 있는지 여부를 알아낸다.
- sdb_update () : 내부 데이터베이스 변경 모듈에 해당되며 동적 갱신 메시지를 검사하여 갱신할 자원레코드를 ldb_update()에 전달하는 기능을 수행한다.
- ldb_update () : 내부 데이터베이스에 직접 자원레코드 구조체를 추가하거나 삭제한다. 또한 NXT와 SIG 자원레코드를 재구성한다.

4. 존 전송의 설계와 구현

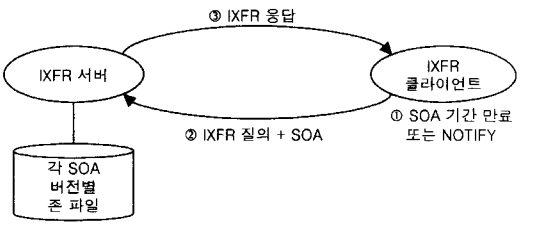
본 장에서는 존 전송을 처리하기 위한 존 전송 요청 모듈과 응답 모듈, 그리고 동적 갱신에 의해 부가적으로 필요한 델타 파일을 입출력하는 모듈의 구체적인 설계와 구현에 대해 설명한다. 또한 전체 존 전송과 점진적 존 전송을 효율적으로 병용하여 사용하는 방식에 대해서도 설명한다.

4.1 존 전송의 설계

안전한 DNS에서는 관리해야할 존 정보가 많기 때문에 점진적 존 전송이 전체 존 전송보다 효율적이다. 하지만 전송해야 할 존 정보의 양이 많을 경우나 기존 DNS와의 호환성 문제 때문에 전체 존 전송방식 역시 지원되어야 한다. 본 논문에서는 이러한 두 가지 존 전송을 효율적으로 병용하여 사용하는 방식에 대해 제시한다.

점진적 존 전송(이후 IXFR)은 변경사항이 있을 경우 기존의 전체 존 전송(이후 AXFR)과는 달리 해당 존의 변경사항만을 대상으로 효율적인 전송을 위해 제안되었다. 따라서 IXFR을 요청하는 클라이언트는 IXFR 서버 즉, 마스터 DNS 서버에게서 해당 존의 변경사항을 받아와 자신의 존을 최신의 내용으로 갱신할 수 있다.

기본적인 IXFR 프로토콜의 개요를 살펴보면 (그림 8)과 같이 IXFR 클라이언트는 자신의 존에서 SOA 자원레코드의 TTL(Time-To-Live)이 기간만료 되거나 NOTIFY 메커니즘[5]에 의해 자신의 존이 갱신이 필



(그림 8) IXFR 서버와 클라이언트간의 상호작용

요하다고 간주한다. 따라서 IXFR 서버에게 점진적 존 전송 요청 메시지를 IXFR 클라이언트의 SOA 자원레코드와 함께 보낸다.

IXFR 서버는 IXFR 클라이언트로부터 받은 SOA 자원레코드의 일련번호로부터 현재까지 변화된 내용만을 추출하여 응답하여야 한다. 이러한 존 정보의 변경은 동적 갱신에 의해 빈번하게 수행되므로 IXFR 서버는 새로운 버전과 이전 버전의 존 파일들 간의 차이점을 계속 간직하고 있어야 한다. 이러한 차이점을 저장하는 파일이 델타파일이며 이를 이용하여 IXFR 클라이언트의 일련번호로부터 현재의 일련번호까지 변경된 존 정보를 정확히 추출해낼 수 있다.

IXFR에 사용되는 질의 메시지의 구조는 기본적인 DNS 질의 메시지와 거의 비슷하다. 즉, 질의 종류가 IXFR이 되고 DNS 메시지의 권한부분에는 클라이언트의 현재 버전의 SOA 자원레코드가 삽입된다. 반면에 IXFR에 사용되는 응답 메시지의 첫 번째와 마지막 부분에는 현재 최신버전의 SOA 자원레코드가 들어가고 그 가운데에 실제로 응답되어질 내용이 포함된다. 점진적 존 전송은 이렇게 기존의 DNS 패킷을 이용하여 서버나 클라이언트의 부가적인 IXFR 처리 모듈만으로 수행이 가능하다. 그러나 이러한 존 전송이 DNS의 가장 핵심 내용인 존 데이터베이스를 변경하는 작업을 수행하므로 이에 대응하는 보안서비스가 필수적으로 필요하게 된다.

보통 DNS가 응답메시지에 전자서명을 해서 데이터의 무결성을 제공하는 반면에 질의 메시지에 대한 전자서명은 많은 경우에 무시되어 진다. 하지만 존 전송의 경우 해당 존의 많은 사항들이 열람이 허가되지 않을 경우에도 모든 사용자들에게 공개될 위험성이 있으므로, 허가된 클라이언트만이 존 전송을 요청하도록 질의 메시지에도 전자서명을 추가하도록 한다. 따라서 이러한 전자서명이 메시지의 무결성과 인증을 제공하게 되어 존 전송에 보안성을 부여하게 된다.

(그림 9)는 점진적 존 전송의 예제를 보여주고 있다. 이 예제에서 IXFR 서버에는 3종류의 일련번호를 가지는 존 정보가 있다. 즉, SOA 자원레코드의 일련번호가 1인 존에서 cs.hongik.ac.kr.이 삭제되어 지고, ce.hongik.ac.kr.인 2개의 자원레코드가 더해져 일련번호 2인 존으로 구성되었다. 또다시 ce.hongik.ac.kr.의 IP 주소 중의 하나가 변경되어 현재 IXFR 서버의 일련번호는 3이 되었다고 가정하자. 이때 IXFR 클라이언트의 SOA 자원레코드는 일련번호가 1이며, 점진적 존 전송을 요청하게 되었을 때의 메시지가 (그림 9)이다.

헤 더	OPCODE=SQUERY, RESPONSE		
질의섹션	QNAME=hongik.ac.kr., QCLASS=IN, QTYPE=IXFR		
응답섹션	hongik.ac.kr.	IN SOA serial=3	
	hongik.ac.kr.	IN SOA serial=1	(삭제)
	cs.hongik.ac.kr.	IN A 203.249.75.5	
	hongik.ac.kr.	IN SOA serial=2	(추가)
	ce.hongik.ac.kr.	IN A 203.249.75.4	
	ce.hongik.ac.kr.	IN A 203.249.75.2	
	hongik.ac.kr.	IN SOA serial=2	(삭제)
	ce.hongik.ac.kr.	IN A 203.249.75.4	
	hongik.ac.kr.	IN SOA serial=3	(추가)
	ce.hongik.ac.kr.	IN A 203.249.75.3	
	hongik.ac.kr.	IN SOA serial=3	
권한섹션	<empty>		
부가섹션	<empty>		

(그림 9) 점진적 존 전송 메시지 예제

하지만 존 전송 메시지의 경우 그 크기가 일반 DNS 메시지에 비해 매우 크므로 효율적인 전송을 위해 다음과 같은 존 전송 메시지 압축 알고리즘을 제시한다.

(그림 9)에서 살펴본 바와 같이 존의 버전을 높이기 위해 필요한 자원레코드들은 두 가지로 분류될 수 있다. 즉 자원레코드를 추가하는 것과 삭제하는 용도의 두 가지이다. 점진적 존 전송 메시지의 압축을 위한 주된 촛점은 바로 추가되는 자원레코드들은 삭제될 자원레코드와는 달리 그 순서가 중요하지 않다는 점에 기인한다. 즉, 이전 버전의 존에서 추가된 자원레코드가 이후에 삭제된다면 그 두 쌍은 상쇄될 수 있고, 남게된 추가될 자원레코드들은 종속성이 없기 때문에 다른 버전에서 데이터베이스에 추가되어도 무관하게 된다. 이렇게 함축된 점진적 존 전송 메시지는 (그림 10)과 같이 된다.

헤 더	OPCODE=SQUERY, RESPONSE	
질의섹션	QNAME=hongik.ac.kr., QCLASS=IN, QTYPE=IXFR	
응답섹션	hongik.ac.kr.	IN SOA serial=3
	hongik.ac.kr.	IN SOA serial=1
	cs.hongik.ac.kr.	IN A 203.249.75.5
	hongik.ac.kr.	IN SOA serial=3
	ce.hongik.ac.kr.	IN A 203.249.75.3
	ce.hongik.ac.kr.	IN A 203.249.75.2
	hongik.ac.kr.	IN SOA serial=3
권한섹션	<empty>	
부가섹션	<empty>	

(그림 10) 함축된 점진적 존 전송 메시지

존 전송에서 또 한가지 고려사항은 존 전송의 방법을 점진적 존 전송만을 사용할 것인가 아니면 점진적 존 전송과 함께 전체 존 전송을 병용하여 사용할 것인가 이다. 앞서 살펴본 바와 같이 존의 변경내용이 많다면 전체 존 전송이 점진적 존 전송보다 효율적이지만, 보통의 경우 존의 변경내용은 매우 적으므로 많은 경우에 점진적 존 전송이 보다 효율적이다.

DNS를 운영하면서 전체 존을 전송할 필요가 있는 경우는 시스템이 초기에 구성될 때와 DNS 시스템을 재구동하는 경우이다. 시스템을 구성하는 경우는 가장 초기에 단 한번밖에 없지만, 재구동은 경우는 많은 이유에서 행하여진다. 즉, 관리자가 수동으로 존의 정보를 갱신한 후에 변경사항을 내부 데이터베이스에 갱신하려면 DNS 시스템을 재구동 하여야 한다. 또한 정전 등의 예기치 않은 사고로 호스트 시스템이 재구동되면, DNS 역시 재구동 되어야 한다. 이러한 요소에 의해 빈번하지는 않지만 전체 존 전송이 필요하게 된다. 또한 점진적 존 전송을 지원하지 않는 DNS와 존 전송을 수행하려면 호환성을 위해 존 전송 역시 구현되어야 한다.

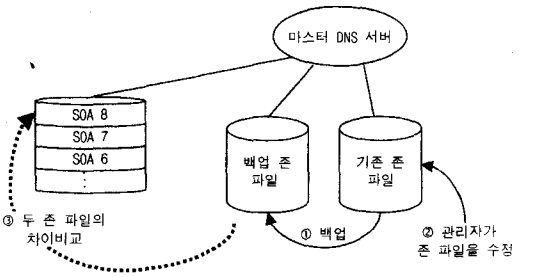
다음은 점진적 존 전송의 효과를 극대화하기 위해 점진적 존 전송만을 사용하는 존 전송 방식과 두 가지 존 전송을 효율적으로 동시에 구현한 방법에 대해 설명하겠다.

4.1.1 점진적 존 전송만을 이용한 존 전송

점진적 존 전송만을 이용하여 존 전송을 수행하는 방법은 존이 가장 처음 구성되었을 경우에만 전체 존 전송으로 존을 슬레이브 DNS 서버에게 전달하고 이후의 변경사항에 대해서는 오직 점진적 존 전송만을 이용하는 방법이다.

이 방법의 장점은 일관된 방법으로 존을 전송하므로 DNS 메시지 전송 규약을 확장시키지 않으며, 전체 존을 재구성하면서 생기는 SIG, NXT 자원레코드 생성 부하를 최소화 할 수 있다. 또한 마스터 DNS 서버와 슬레이브 DNS 서버의 통신량도 전체 존 전송에 비해 현저히 줄어들게 되므로 상당한 장점을 지니게 된다.

하지만 이렇게 점진적 존 전송만을 사용하는 경우 발생하는 문제점들도 적지 않다. 가장 큰 문제점은 관리자에 의해 수동으로 존을 갱신하는 경우 이전 버전의 존 정보와의 차이점을 알 수 없으므로 델타파일을 생성하는 것이 불가능하다는 것이다. 즉, DNS 갱신의 내용은 모두 순차적으로 델타파일에 유지되어야 하지만, 관리자가 수동으로 존을 갱신하고 DNS를 재구동하는 시점은 예측할 수 없기 때문에 그 시점에서는 델타파일을 생성할 수 없게 되어 점진적 존 전송을 할 수 없는 문제점이 발생한다.



(그림 11) 관리자에 의한 존 정보의 변경

이를 해결하는 방법은 (그림 11)과 같이 모든 갱신 후 마다 현재 존의 내용을 존 파일과 함께 백업 파일에도 저장하는 것이다. 따라서 관리자에 의해 수동으로 존 파일을 수정하더라도 수정하기 이전의 존 파일이 백업되어 있으므로 이 두 개의 존 파일을 비교하여 델타파일을 생성할 수 있게 된다.

점진적 존 전송만을 사용할 때 발생되는 다른 문제점은 존 정보의 변화량이 많을 경우에 전체 존 전송보다 더 큰 메시지를 응답하게 되는 경우도 발생하게 된다는 점이다.

4.1.2 점진적 존 전송과 전체 존 전송을 병행 이용

점진적 존 전송의 경우에도 존을 처음 구성한 경우 전체 존을 한꺼번에 전송하는 매커니즘이 필요하다. 따라서 전체 존 전송을 완전히 배제한 구현은 적당치 못할 것이다. 더욱이 다른 DNS 간의 호환성을 중요시한다면 점진적 존 전송과 전체 존 전송을 효율적으로 동시에 운용하는 것이 좋은 선택이 될 것이다.

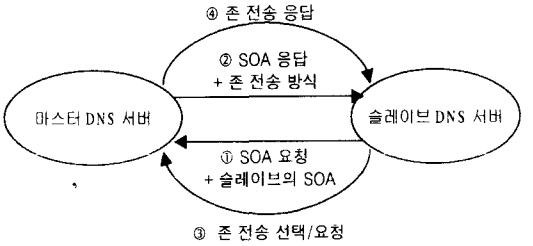
이미 살펴본 바와 같이 전체 존 전송은 점진적 존 전송과는 약간 다른 방법으로 존을 전송하게 된다. 즉, 전체 존 전송은 (그림 2)와 같이 SOA 자원레코드의 갱신 필드(refresh field)에 있는 값이 감소하여 0이 되었을 때 슬레이브 DNS 서버가 마스터 DNS 서버에게 SOA를 요청하게 된다. 그리고 슬레이브는 자신의 SOA 일련번호와 응답 받은 SOA 자원레코드 사이의 내용을 비교하여 전체 존 갱신을 요청할 지 여부를 결

정한다. 이 방법은 총 4번의 메시지가 교환되어야 존 전송을 마칠 수 있게 된다. 하지만 점진적 존 전송은 (그림 3)과 같이 슬레이브 DNS의 점진적 존 전송 요청을 보고 마스터 DNS 서버가 존 전송 여부를 판단하게 된다.

점진적 존 전송은 전체 존 전송과는 달리 갱신 필드의 값을 이용하지 않고 SOA 자원레코드의 TTL 값을 사용하게 된다. 따라서 서로 다른 값을 가지고 주기적으로 전체 존 전송과 점진적 존 전송을 반복하게 된다. 보통 갱신 필드와 TTL값이 일치하지는 않지만 간격이 비슷할 경우가 있으므로 이렇게 두 존 전송 방식을 병용하게 된다면 짧은 기간동안 두 존 전송 방법을 모두 사용하는 비효율을 가져올 수 있다.

따라서 본 논문에서는 기존 DNS와 호환이 가능할 뿐 아니라 효율적인 존 전송의 병행을 위해 기존 두 방식을 통합한 존 전송 혼용 방식을 제안한다. 즉, 두 존 전송 방식의 장점만을 취하여 관리자에 의한 존 갱신 등의 존 정보의 많은 변화가 있을 때는 전체 존 전송을 사용하고 동적 갱신의 경우에는 변경되는 정보의 양이 그리 크지 않으므로 점진적 존 전송을 사용하도록 한다.

이러한 동작이 가능하려면 두 존 전송의 요청방식을 통일하여야 한다. 하지만 두 방식을 통일하게 된다면 다른 DNS와 호환이 되지 않는 문제점을 야기시킬수 있다. 따라서 기존 DNS에서도 문제가 되지 않도록 DNS 메시지의 부가적 섹션(Additional Section)에 존 전송을 선택할 수 있도록 하는 별도의 정보를 포함하여 질의하도록 프로토콜을 수정하였다.



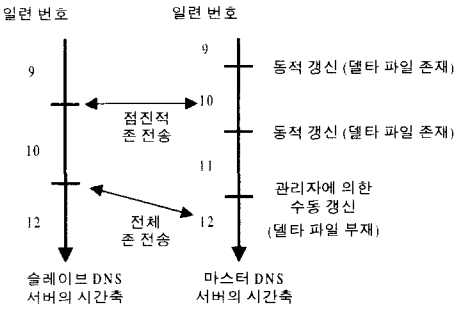
(그림 12) 전체 존 전송과 점진적 존 전송의 혼용방식

(그림 12)에서는 전체 존 전송과 점진적 존 전송의 혼용방식에 대해 설명하고 있다. 즉, IXFR 클라이언트가 전체 존 전송의 방식과 동일하게 IXFR 서버의 SOA 자원레코드를 요청한다. 이때 전체 존 전송 방식과 다른 점은 SOA 요청 메시지의 부가적 섹션에 현재 존의 버전번호를 추가하여 전송한다는 점이다. 따라서 IXFR 클라이언트의 버전번호를 알 수 있게된 서버는 델타파일에서 해당 버전의 존 정보가 있는지 확인하여, 만약 델타파일이 존재 한다면 점진적 존 전송을 사용할 것이고, 그렇지 않다면 전체 존 전송을 선택하게 된다. IXFR 서버가 선택한 존 전송 방식을 IXFR 클라이언트에게 전달하기 위해서는 SOA 응답 메시지의 부가적 섹션에 서버의 SOA 자원레코드를 추가하는 것이다. 따라서 이 메시지를 받게 되는 IXFR 클라이언트의 경우 부가적 섹션에 SOA 자원레코드가 존재한다면 점진적 존 전송 요청 메시지를 서버에 보낼 것이고, 그렇지 않을 경우 전체 존 전송 메시지를 서버에 요청할 것이다.

존 전송 방식을 선택하기 위해 SOA 자원레코드를 부가적 섹션에 추가하는 이유는 기존 DNS와의 호환성을 제공하기 위해서 이다. 부가적 섹션의 용도가 질의 또는 응답 이외에 필요한 정보를 제공하는 것이므로 기존 DNS 에서는 부가적 섹션의 자원레코드를 검색하려 하지 않을 것이다. 따라서 무시되는 부가적 섹션의 자원레코드를 제외한다면 전체 존 전송의 방식과 동일하게 되고 따라서 (그림 12)의 어느 한 서버가 점진적 존 전송을 이해하지 못하는 기존의 DNS라면 단순히 전체 존 전송방식이 선택되어 호환성을 유지할 것이다.

안전한 DNS에서 마스터 DNS 서버가 전체 존 전송 방식을 선택하는 경우는 관리자에 의한 존 정보의 변경 등 전송해야 할 존 정보가 많을 경우이다. 반면에 점진적 존 전송은 동적 갱신에 의해 자동적으로 갱신된 정보를 슬레이브 DNS 서버로 전송하는 것이다. (그림 13)은 이러한 두 가지 존 전송 방식이 혼용되어 사용되는 예를 설명한다. DNS 서버의 초기 SOA 버전은 9이고, 2번의 동적 갱신에 의해 11이 되었다고 가정하자. 이후 관리자에 의해 수동으로 존을 갱신하게 되어 델타파일이 만들어지지 않게 되고, 따라서 점진적 존 전송은 불가능하게 된다. 첫 번째 존 전송은 마스터 DNS 서버의 SOA 버전이 10이고, 슬레이브 DNS의 SOA 버전이 9일 때 발생했다. 이때는 점진적 존 전송에 의해 SOA 버전 9와 SOA 버전 10의 차이만을 델타파일에서 찾아 전송하면 된다. 두 번째 존 전송은 마스터 DNS 서버의 SOA 버전이 12일 때 발생한다. 존 전송을 요청한 슬레이브 DNS의 현재 SOA 버전이 10이므로 SOA 버전 10과 SOA 버전 12의 차이만을

전송하면 된다. 하지만 SOA 버전 11과 SOA 버전 12 사이의 델타파일이 존재하지 않으므로 이러한 변경사항만을 전송하는 것이 불가능하게 된다. 따라서 전체 존 전송을 이용하여 존 전송을 수행하게 된다.



(그림 13) 존 전송 혼용방식의 예제

4.1.3 델타 파일의 입출력 모듈 설계

델타파일은 이미 언급한 바와 같이 동적 갱신과 점진적 존 전송을 위해 새로 추가된 자료구조로서 DNS 메시지 형태로 파일에 저장되게 된다. 이후에 점진적 존 전송 요청을 받게 된다면, 보안 검사를 수행하고 전송된 SOA 자원레코드의 일련번호를 조사하여 해당 존 이름의 델타파일에서 갱신될 자원레코드를 찾게 된다.

현재 존의 SOA 자원레코드	SOA 자원레코드 + 변경된 자원레코드 집합	SOA 자원레코드 + 변경된 자원레코드 집합	....	현재 존의 SOA 자원레코드

(그림 14) 점진적 존 전송의 응답

점진적 존 전송의 응답은 (그림 14)과 같이 메시지의 시작과 끝 부분에 구분자 역할을 하도록 마스터 DNS 서버의 현재 SOA 자원레코드를 하나씩 위치시키고 그 사이에는 요청된 버전의 존 정보로부터 현재 버전의 존 정보까지 변화된 내용을 포함한다. 존 정보의 변경사항은 변경된 시기의 버전을 나타내는 SOA 자원레코드와 그 버전에서 변경된 자원레코드들의 집합으로 구성되어 진다. 따라서 델타파일은 이러한 변경사항들은 순차적으로 구성하여 파일에 저장하여야 한다.

델타파일의 입력모듈은 현재 존의 버전을 나타내는 SOA 자원레코드와 함께 변경된 자원레코드의 집합을 해당 파일의 마지막 부분에 이어 쓰는 작업을 수행하여 갱신된 존 정보를 순차적으로 저장한다. 델타 파일의 출력모듈은 SOA 자원레코드의 일련번호를 입력으로 델타파일의 처음 부분부터 해당 버전의 SOA 자원레코드를 찾는다. 만약 해당 SOA 자원레코드를 델타파일에서 찾게 된다면 변경된 자원레코드 집합을 DNS 메시지 형태로 응답하게 된다.

4.2 존 전송의 구현

존 전송의 기능을 수행하기 위한 모듈들은 존 전송 방법 선택 모듈, 존 전송 요청/응답 모듈, 그리고 마지막으로 델타 파일의 입출력 모듈로 구성되어 진다. 위의 3가지 모듈들의 구현에 대해 자세히 설명하도록 하겠다.

4.2.1 존 전송 방법 선택 모듈의 구현

앞서 설명한 바와 같이 존 전송 방식의 선택은 전체 존 전송과 점진적 존 전송의 방식을 혼용하여 구현되었다. 따라서 존 전송의 방식을 선택하는 주체는 마스터 DNS 서버가 될 것이며, (그림 15)와 같이 구현된다. 존 전송 방식을 선택하기 위해 델타파일에서 갱신 요청자의 일련번호가 존재하는지 검색하여 만약 델타파일에 요청된 일련번호가 있다면 점진적 존 전송을 수행할 수 있는 상태가 되므로 부가적 섹션에 마스터 DNS의 SOA 자원레코드를 삽입하여 응답하게 된다. 이 메시지를 응답 받은 슬레이브 DNS 서버는 메시지를 확인하여 만약 SOA 자원레코드가 부가적 섹션에 존재한다면 점진적 존 전송 방식을 선택할 것이고, 반대로 부가적 섹션에 SOA 자원레코드가 없다면 전체 존 전송을 요청하게 된다.

```
serial = 슬레이브 DNS의 SOA의 일련번호
존 전송 방식 = get_serial(존의 이름, serial);
if (존 전송 방식 == M_IXFR)
    Add_Additional(M, 마스터 DNS의 SOA 자원레코드);
Make_Message(msg, M);
```

(그림 15) 존 전송 선택모듈의 구현

4.2.2 존 전송 요청/응답 모듈의 구현

전체 존 전송 요청 모듈은 리졸버 함수인 res_mkquery()를 사용하여 T_AXFR(IETF에서 252로 지정)[3]을 요청한다. 이때 응답해야 할 자원레코드들은

모두 응답 섹션에 포함되기에 너무 많은 내용을 가지고 있으므로, 한 개의 자원레코드마다 하나의 DNS 응답 메시지를 만들어서 수신과 송신을 마지막 자원레코드까지 계속하여야 한다.

이렇게 응답된 자원레코드들은 하나씩 분석되어 해당 존 파일에 순차적으로 저장되어 진다. 마지막 자원레코드는 맨 처음에 보내진 SOA 자원레코드가 중복되어 보내지므로 존 전송의 마지막임을 알 수 있다. 따라서 존 전송이 완료된 후에는 존 파일에 갱신된 내용이 저장되어 있게 된다. 관리자는 변경된 내용을 내부 데이터베이스에 갱신하려면 DNS에 유닉스 HUP 시그널을 보내어 단순히 재구동 하면 된다.

전체 존 전송 응답 모듈은 요청 모듈에게 존 정보의 해쉬 포인터를 시작으로 모든 존의 정보를 하나씩 전송하는 작업을 수행한다. 응답 모듈은 무한 반복문으로 자원 레코드를 받게 되므로 모든 응답의 끝을 알리기 위해 존 전송의 시작과 마지막 부분에 현재 존의 SOA 자원레코드를 응답하게 된다.

점진적 존 전송 요청 모듈은 대부분 전체 존 전송 모듈과 동일하다. 즉 res_mkquery()를 사용하여 T_IXFR (251로 지정)[3] 메시지를 요청한다. 응답 받게 되는 자원레코드들은 모두 DNS 메시지의 응답 섹션에 포함되어 있으므로 동적 갱신의 내부 데이터베이스 변경 모듈을 호출하여 직접 내부 데이터베이스에 갱신하고 갱신된 내용을 존 파일에 저장하는 작업을 수행한다. 물론 보안 기능을 제공하기 위해 SIG와 NXT 자원레코드를 재구성하는 작업이 뒤따른다. 이와 같이 점진적 존 전송 방식은 전체 존 전송 방식과는 달리 직접 존의 데이터베이스를 갱신하므로 DNS 시스템을 재구동 하는 별도의 작업은 필요 없게 된다. 하지만 변경된 존의 정보를 존 파일에 저장하여 혹시 발생할 수 있는 시스템의 재구동에 대비하여야 할 것이다.

점진적 존 전송의 응답 모듈은 요청된 SOA의 일련번호부터 현재까지의 변경된 존 정보를 델타 파일 처리 모듈에서 DNS 표준 메시지 형태로 응답 받게 된다. 따라서 이러한 메시지를 단순히 존 전송 메시지의 응답 섹션에 포함시켜 존 전송 요청자인 슬레이브 DNS 서버에 전송하면 된다.

마지막으로 보안서비스를 제공하기 위해 고려할 사항은 비교적 간단하다. DNS 표준 메시지와 동일한 형식의 메시지를 사용하기 때문에 메시지를 파싱하는 순간에 모든 자원레코드에 종속된 SIG 자원레코드를 검증하고, 메시지 인증도 성공적으로 수행한 후에야 점진적 존 전송이 이루어져야 할 것이다.

4.2.3 델타 파일 입출력 함수의 구현

델타 파일의 입력 함수는 내부 데이터베이스의 변경을 요청한 DNS 메시지의 갱신부분만 추출하여 해당 존의 이름 뒤에 확장자(".delta")가 더해진 이름의 델타 파일의 마지막 부분에 덧붙이는 형태로 구현된다. 보통 한 개의 DNS 서버가 하나의 존을 관리하는 형태로 구성되지만 실제로는 DNS 서버 하나가 여러 개의 존을 관리할 수 있도록 구현되어 있다. 따라서 각 존에 해당되는 존 파일의 이름이 다른 것과 마찬가지로 델타 파일 역시 여러 개의 존에 각각 하나씩 존재하여야 함으로 구분이 쉽도록 존의 이름을 사용하여 델타 파일을 구성하도록 한다.

반면에 델타 파일의 출력 함수는 갱신 요청 메시지에 있는 SOA 일련번호를 입력으로 해당 델타 파일에서 자원레코드를 찾는다. 만약 요청된 SOA 자원레코드의 일련번호가 델타파일의 가장 첫 부분에서 발견되었다면 그 이후의 자원레코드를 내부 데이터베이스 형식으로 하나씩 리턴하고 그렇지 않을 경우 일치하는 두 번째 SOA를 자원레코드가 발견된 이후의 자원레코드를 리턴 하여야 한다. 이러한 이유는 (그림 9)와 같이 중간 부분의 SOA 자원레코드는 추가와 삭제 부분으로 나뉘어져 있기 때문이다.

4.2.4 구성 함수별 세부 구현사항

존 전송 모듈을 구성하는 함수들은 아래와 같다. 즉, 존 전송의 선택부터 선택된 존 전송요청을 일률적으로 수행하는 존 전송 API 함수와 이를 보조하는 존 전송 선택 함수, 전체 존 전송 함수, 점진적 존 전송 함수, 그리고 델타파일 입출력 함수들로 구성된다.

- request_zone() : 존 전송 요청을 위한 API이다. 즉, 두 가지 존 전송 방식 중 하나만을 선택하여 마스터 DNS서버에게 질의하는 함수로 존 전송에 관계되는 모든 함수를 이 함수가 호출하는 형태로 구성된다.
- query_soa() : 존 전송을 위해 SOA 자원레코드를 마스터 DNS 서버에게 질의한다.
- resp_soa() : SOA 자원레코드를 존 트랜스퍼를 위한 응답으로 슬레이브 DNS 서버에게 응답한다. query_

soa()와 마찬가지로 존 전송 선택 모듈에 해당된다.

- request_axfr() : AXFR 질의를 생성하여 마스터 DNS 서버에 전송하고, 존 전송을 응답 받는다. print_zone() 이용하여 존 파일에 저장한다.
- request_ixfr() : IXFR 질의를 생성하여 마스터 DNS 서버에 전송하고, 존 전송을 응답받는다. 메시지의 응답섹션에서 자원레코드들을 하나씩 읽어와 ixfr_dbupdate()에 전달하여 내부 DB에 동적으로 갱신한다.
- resp_ixfr() : IXFR 질의를 바탕으로 응답메시지를 만들어 슬레이브 DNS에게 전송한다.
- ixfr_dbupdate() : IXFR 응답 메시지의 응답섹션에 포함된 자원레코드들을 내부 DB에 ldb_update()를 통해 갱신한다.
- get_delta() : 델타 파일에서 해당 일련번호 이후에 동적 갱신된 자원레코드들을 메시지 형태로 저장하여 돌려준다.
- put_delta() : 자원레코드를 메시지 형식으로 변환하여 델타파일에 저장한다.

5. 확장된 안전한 DNS의 활용 및 실험

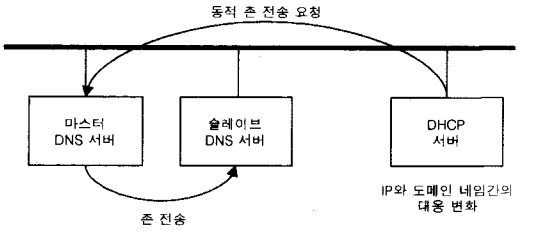
본 장에서는 구현된 확장된 DNS를 다른 인터넷 응용 서비스와 연계하여 사용할 수 있는 방안에 대해 제시하고 구현된 DNS의 실험 결과에 대해 설명한다.

5.1 확장된 DNS의 활용 방안

동적 갱신과 존 전송 기능이 보안서비스와 함께 제공되는 확장된 DNS는 인터넷 보안에 크게 기여할 뿐 아니라, 다른 인터넷 응용서비스와의 접속이 가능하게 된다. 즉 이미 전세계에 광범위하게 분포되어 있고 그 사용빈도도 매우 높은 DNS를 이용하여, DHCP 서버나, CA 서버 또는 멀티캐스트 에이전트 등과 연계한다면 그 효율성은 극대화 될 것이다.

5.1.1 DHCP 서버

DHCP는 한정된 IP를 여러 호스트에 할당하기 위한 프로토콜이다. 즉, 통신 사업자들이 많은 사용자들에게 일일이 IP를 할당하는 것이 비효율적이기 때문에 이러한 프로토콜을 이용하여 사용자가 접속할 때마다 DHCP 서버가 자동으로 IP를 할당하게 된다. 하지만 DHCP 서버에 접속하는 호스트들은 수시로 변경되는 IP 주소와는 무관하게 도메인 이름은 변하지 않기를 원한다. 따라서 DHCP 서버에 의해 IP 주소가 변경될 때마다 DNS 서버는 호스트의 도메인 이름이 변경되지 않도록 할당된 IP 주소와 도메인 이름을 DNS의 내부 데이터베이스에 갱신하여야 한다. 이러한 자동갱신은 동적 갱신에 의해 효율적으로 수행된다. (그림 16)은 DHCP와 DNS 서버간에 동적 전송과 점진적 존 전송을 이용한 구성을 나타내고 있다. DHCP 서버는 수시로 변경되는 IP와 도메인 간의 대응관계를 자동적으로 DNS 메시지로 변환하여 마스터 DNS 서버에 갱신을 요청한다. 슬레이브 DNS 서버는 존 전송을 요청하여 마스터의 변경된 내용과 일치하는 존 정보를 유지하게 된다.



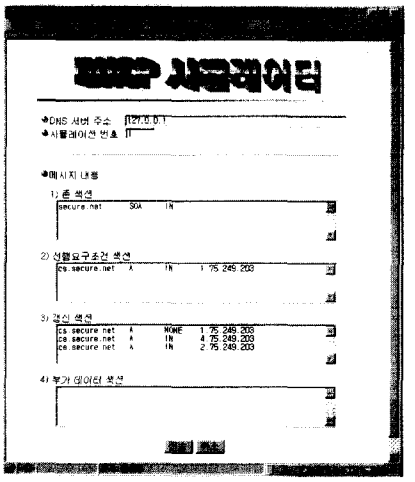
(그림 16) DNS와 DHCP 서버간 동작

5.1.2 인증서 서버

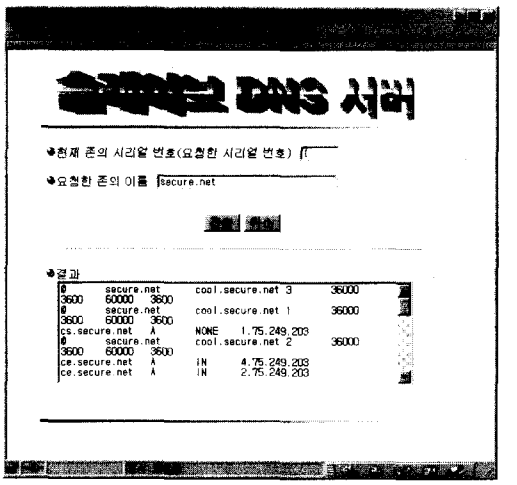
인증서 서버(Certificate Authority Server)도 마찬가지로 여러 사용자의 공개키에 대한 인증서를 제공하고 또한 자신의 공개키와 사용자들의 공개키를 분배하는 역할을 수행한다. 따라서 DNS 서버가 공개키와 인증서의 저장장소로 사용된다면 이러한 인증서 서버와의 통신으로 자동적으로 KEY와 SIG 자원레코드로 변환하는 기능을 수행하는 것이 효율적일 것이다. 이 때 발생하는 문제점은 수많은 공개키와 인증서가 발행되고 기한이 만료되면 다시 폐기되는 상황이 발생하게 되므로 역시 DHCP 서버의 경우와 마찬가지로 존의 갱신이 동적으로 수행되어야 할 것이다.

5.1.3 멀티캐스트 에이전트

멀티캐스트 주소는 일반 IP와는 다르게 수시로 바뀌게 되며 멀티캐스트 에이전트가 멀티캐스트 도메인 네임에 IP 그룹 주소를 동적으로 할당한다. 이렇게 할당된 주소는 이전에 할당되었던 주소와 다르게 되므로 도메인 이름을 그대로 사용하려면 DNS 서버에 변경사항을 등록하여야 한다. 따라서 자동으로 존의 정보가 갱신되는 동적 갱신과 함께 변경사항이 해당 존으로 전송될 수 있도록 존 전송을 지원하여야 한다.



(그림 17) 웹 기반의 DHCP 시뮬레이터



(그림 18) 웹 기반의 슬레이브 DNS 서버

5.2 확장된 DNS의 실험

구현된 확장된 안전한 DNS를 점검하기 위하여 DHCP 서버와의 연동 실험을 수행하였다. 실험 환경은 (그림 16)과 같이 웹 기반의 DHCP 시뮬레이터와 마스터 DNS 서버, 그리고 슬레이브 DNS 서버로 구성되어진다. 즉, DHCP 시뮬레이터에서는 IP와 도메인 이름이 변경될 경우 동적 갱신 요청을 수행하며, 요청된 메시지의 내용을 (그림 17)과 같이 웹 페이지(WEB Page)에서 확인할 수 있다. 마스터 DNS 서버는 DHCP 서버에서 요청된 동적 갱신 메시지를 처리하고, 변경된 내용을 슬레이브 DNS 서버의 요청에 의해 존 전송을 수행한다. 따라서 슬레이브 DNS 서버는 DHCP 서버에 의해 변경된 데이터베이스의 내용을 마스터 DNS 서버에게 전달받아 최종적으로 (그림 18)과 같은 결과를 얻게 된다.

6. 결 론

인터넷 기반 구조인 DNS를 다른 인터넷 응용프로그램과 연동하려면 보안 서비스를 제공하는 것 이외에 효율적인 존 정보의 갱신 기능을 제공하여야 한다. 동적 갱신 기능은 동적으로 존 정보를 추가 또는 삭제할 수 있도록 하여 효율적인 존의 관리를 가능하게 하였다. 또한 갱신된 존 정보는 하위 DNS 서버로 전달되어야 하므로 효율적인 존 전송 기능을 제공하는 것이 중요하다.

이에 따라 본 논문에서는 이미 개발된 안전한 DNS를 기반으로 보안 서비스가 추가된 동적 갱신과 점진적 존 전송 방안에 대해 분석하였으며 이에 따른 설계와 구현을 수행하였다. 또한 효율적인 존 전송을 위해 기존의 두 존 전송을 통합하는 알고리즘과 함께 존 전송 메시지의 압축 알고리즘을 제시하였고, 존 전송과 동적 갱신을 통합하기 위해 델타파일이라는 새로운 자료구조를 제시하였다. 그리고 구현된 시스템을 점검하기 위해 실험을 통하여 증명하였다.

이러한 기능들이 추가된 안전한 DNS 서버는 본래의 기능인 네이밍 서비스 이외에 인증서 서버와 연동하여 공개키의 분배와 인증서의 분배 등의 서비스에 이용이 가능할 뿐 아니라 구축중인 국가 초고속 인터넷 사업의 보안을 위한 기본 요소로 활용할 수 있을 것으로 기대된다.

참 고 문 헌

[1] RFC 1032 : M.Stahl, "Domain administrators guidc," 1987. 11.

[2] RFC 1033 : M.Lottor, "Domain administrators operations guide," 1987. 11.

[3] RFC 1183 : C. Everhart, L. Mamakos, R. Ullmann, P. Mockapetris, "New DNS RR Definitions," 1990. 10.

[4] RFC 1995 : M. Ohta, "Incremental Zone Transfer in DNS," 1996. 8.

[5] RFC 1996 : P. Vixie, "A Mechnism of Prompt Notification of Zone Changes (DNS NOTIFY)," 1996. 8.

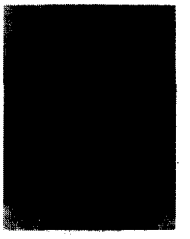
[6] RFC 2136 : D. Eastlake, "Dynamic Updates in the Domain Name System (DNS UPDATE)," 1997. 1.
[7] RFC 2137 : D.Eastlake, "Secure Domain Name System Dynamic Updates," 1997. 4.
[8] RFC 2535 : D.Eastlake, "Domain Name System Security Extensions," 1999. 3.
[9] 심영철 외 4, "확장된 DNS 보안 메커니즘의 설계 및 구현", 한국정보처리학회 논문지 제6권 제1호, 1999. 1.
[10] C.S. Im, O.H. Byeon, Y.C. Shim, "Using DNS as a Certificate Repository in the Public Infrastructure," IASTED Int.Conf. on Parallel and Distributed Computing and Networks, 1998. 12.
[11] Y.C. Shim et al, "Extension and Design of Secure Dynamic Updates in Domain Name Systems," APCC/OECC'99.

심 희 원

e-mail : hwshim@cs.hongik.ac.kr
1998년 단국대학교 전자계산학과(학사)
1998년~현재 홍익대학교 전자계산학과 석사과정
관심분야 : 분산 병렬처리, 네트워크 보안, 멀티 캐스팅

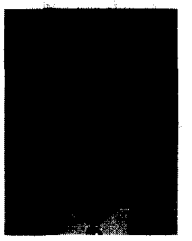
심 영 철

e-mail : shim@cs.hongik.ac.kr
1979년 서울대 전자공학과(학사)
1981년 한국과학기술원 전기 및 전자과(석사)
1991년 University of California, Berkeley 전산학(박사)
1981년~1984년 삼성전자 대리
1991년~1993년 University of California, Berkeley 연구원
1993년~현재 홍익대학교 정보컴퓨터공학부 부교수
관심분야 : 분산 병렬처리, 인터넷 프로토콜, 네트워크 보안, 망관리



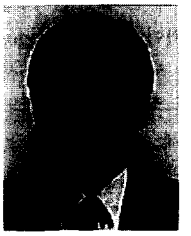
임 찬 순

e-mail : csim@garam.kreonet.re.kr
1994년 중앙대학교 전자계산학과 졸업(공학사)
1996년 중앙대학교 컴퓨터공학과 대학원 졸업(공학석사)
1996년~1999년 한국전자통신연구원 슈퍼컴퓨터센터 연구원
1999년~현재 연구개발정보센터 슈퍼컴퓨팅사업단
관심분야 : 망 관리, 인터넷 보안, 방화벽 시스템



이 만 희

e-mail : mhlee@garam.kreonet.re.kr
1995년 경북대학교 컴퓨터공학과 (학사)
1997년 경북대학교 컴퓨터공학과 (석사)
1997년~1999년 한국전자통신연구원 슈퍼컴퓨터센터 연구원
1999년~현재 연구개발정보센터 슈퍼컴퓨팅사업단
관심분야 : 병렬알고리즘, 망관리, 네트워크 보안



변 옥 환

e-mail : ohbyeon@garam.kreonet.re.kr
1979년 한국항공대학교 통신정보공학과 졸업 (공학사)
1985년 인하대학교 전자공학과 대학원 졸업 (공학석사)
1993년 경희대학교 전자공학과 대학원 졸업 (공학박사)
1978년~1998년 시스템공학연구소
1998년~1999년 한국전자통신연구원 책임연구원
1999년~현재 연구개발정보센터 슈퍼컴퓨팅사업단
관심분야 : 망 관리, SNMP, 인터넷 보안, 방화벽 시스템