DNS장애 발생 시 효율적인 대처방안

Efficient Management of DNS Failure

임양원, 임한규
안동대학교 멀티미디어공학과

Yang-Won Lim(ishamaim@hanmail.net), Hankyu Lim(hklim@andong.ac.kr)

요약

DNS는 인터넷주소 자원관리의 핵심이며 가장 기본이 되는 네이밍 서비스를 제공한다. 현재의 DNS는 트리구조로 계층화 되어있다. 하지만, 상위 DNS의 장애발생으로 인한 하위 DNS는 정상적인 연결이 어렵게 되며, 보조 DNS를 운영한다고 하더라도 그 위험성은 여전히 존재하게 된다. DNS는 빠른 검색이 가능하다는 장점 때문에 지금의 계층 구조를 버릴 순 없다. 본 논문에서는 이와 같은 DNS 장애 발생 시 효율적으로 대처할 수 있도록 지금의 계층구조의 장점을 유지하고 수평적이면서 독립적인 DNS구조를 추가해서 장애가 발생하더라도 임시적으로 로컬 DNS의 운영이 가능하도록 하였다.

■ 중심어 : | DNS | Domain Name System | DNS 장애 | Local DNS |

Abstract

The Domain Name System (DNS) is the core system for managing Internet address resources, providing the most fundamental naming service. Currently, the DNS is classified into a tree structure. In this structure, it is difficult to normally access to the lower DNS, when there is an error in the upper DNS. Such a risk still remains even when a supplementary DNS is operated. However, due to the merit of the DNS enabling fast searches, it is impracticable to abandon the current tree structure. To efficiently correspond to DNS errors, this study suggests a method where the merit of the current tree structure is kept, while a temporary operation of the local DNS is available when errors occur by adding a horizontal and independent DNS structure.

■ keyword : | DNS | Domain Name System | DNS Failure | Local DNS |

I. 서 론

최근 정보통신 기술의 발달과 더불어 인터넷의 보급과 활용이 일반화되면서 회사, 관공서, 가정에서도 초고속 인터넷을 활용하여 정보의 교류와 금융, 전자상거래, 게임 등 정보의 활용이 일상 생활화 되었다[1]. 초기에는 네트워크에 연결된 호스트이름과 이에 대응되는 IP 주소를 입력하는 것으로도 충분하였으나, 네트워크가 급격히 확장되면서 모든 호스트가 NIC에 접속하여 호스트파일을 받는다는 것은 심각한 문제를 야기하게 되

* 본 논문은 2006학년도 안동대학교 학술연구 조성비에 의하여 연구되었습니다.

접수번호 : #070827-001
접수일자 : 2007년 08월 27일
심사완료일 : 2007년 09월 28일
교신저자 : 임한규, e-mail : hklim@andong.ac.kr

고, NIC에서 호스트파일을 관리한다는 것은 불가능하게 되었다. 이런 문제점을 보완하기 위해 나온 것이 DNS이며 지금은 계층적 네임스페이스를 이용하여 네트워크 확장에 대비한 확장성을 제공하고 분산관리를 지원하기 때문에 상당히 효율적인 방법이다[2]. 이러한 네트워크의 발전과 더불어 사이버 공격의 피해도 해마다 급증하고 있다. 특히, DNS서버가 공격의 대상이 되기 쉬우며 DNS서버가 피해를 입게 되면 사용자들은 인터넷을 사용할 수 없게 되어 큰 혼란을 가져오게 된다[3].

일반적으로 DNS에 관한 정보는 각각의 인트라넷의 상위 DNS서버에서 정보를 보유하고 있으나, 대부분 상위 DNS서버는 자신의 하위 네트워크의 DNS정보만을 보유하고 있어, 클라이언트가 찾는 사이트의 DNS 정보가 없을 경우가 대부분이고, 그러한 경우 다시 상위 DNS 서버로 넘겨 인터넷 접속을 가능하게 한다[4]. 따라서 지금의 수직적인 계층구조를 가진 DNS망에서는 자연히 상위 DNS에 많은 접속이 이루어진다. 이와 같이 과부하나 해킹 등으로 상위 DNS서버가 마비될 경우 그 하위에 있는 DNS는 사이트 주소를 찾을 수 없어 인터넷이 마비, 지연되는 장애가 발생한다는 것이다[5]. 지금의 DNS는 수직적인 계층구조가 가지는 빠른 검색과, 효율적인 자료의 관리 등 많은 장점이 있지만, 또한 연결고리가 불안하고, 중요 노드의 장애 발생시, 그 하위에 있는 많은 클라이언트가 인터넷주소(IP Address)를 해석하지 못하는 사태는 언제든지 일어날 수 있다[6].

항상 공격과 위험에 노출된 지금의 DNS를 단점과 장애로 인해 모든 인터넷 접속이 마비되는 최악의 상황을 대비할 수 있는 안전장치를 마련할 수 있다면, 피해를 최소로 줄일 수 있을 것이다.

따라서 본 논문에서는 이와 같은 DNS가 가지고 있는 문제점에 대한 개선방안으로 지금의 계층구조의 효율을 최대한 이용하면서, 장애를 대비해서 독립적 DNS와 수평적 구조를 추가하여 병행 운영할 수 있는 대처방안을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 현재 운영중인 DNS의 구조와 동작원리를 살펴보고, DNS의 장애의 원인이 될 수 있는 공격과 장단점을 살펴보고 논의해본다. 3장에서는 DNS 장애로 인한 인터넷 망의 피해를 최소화하기 위해, 본 논문에서 제안하는 DNS장애 발생 시 대처방안을 위한 시스템을 설명하고, 4장에서는 구현 및 동작원리를 설명하고, 끝으로 5장에서는 결론을 맺는다.

II. 관련 연구

본 장에서는 현재 사용하고 있는 계층적 구조의 DNS의 발전과정과 이름해석의 동작원리 및 구성요소에 대하여 살펴보고, DNS장애의 주요 원인이 되는 해킹 방법과 계층적 구조의 장, 단점 및 DNS 장애에 따른 피해사례를 살펴본다.

1. DNS(Domain Name System) 개요

인터넷을 통하여 컴퓨터가 서로 통신을 하려면 먼저 각 컴퓨터는 논리적으로 할당된 인터넷주소(IP Address)가 있어야 한다. 정보를 요청하는 클라이언트(Client)도 인터넷주소가 필요하지만, 정보를 제공하는 서버도 같은 이유로 인터넷주소가 필요하다. 이러한 인터넷주소는 편의상 32비트 숫자로 표현되기 때문에 사용자(Client)가 일일이 전부 기억한다는 것은 어려운 것이 사실이다. 그래서 사용자가 기억하기 쉽도록 이름을 부여하여 사용한다. 그래서 사람을 위한 이름과 컴퓨터가 사용하는 인터넷주소를 서로 연결(Mapping)하는 장치가 바로 DNS이다[7].

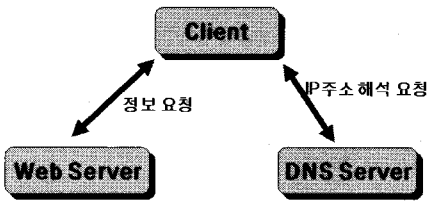


그림 1. IP주소해석의 기본구조

[그림 1]은 인터넷 주소해석의 기본구조를 나타낸 것이다. 정보를 제공하는 웹 사이트는 저마다 해당 IP주

소를 가지고 있다. 네트워크상의 컴퓨터로 전 세계 웹 서버 중 어느 하나에 접속하려면, 그 서버의 IP주소를 알고 있어야 한다. IP주소를 알아야만, 온 세계를 돌며 웹 서버를 찾아 접속하는 복잡한 라우팅 절차를 시작할 수 있다. 두 개의 컴퓨터가 서로 통신을 하려면, 먼저 컴퓨터의 네트워크 카드는 서로 알고 있어야 한다. ARP(Address Resolution Protocol)가 이들 IP주소를 MAC(Medium Access Control)주소로 변환하는 기능을 수행한다. IP주소들은 호스트명으로 사용하는 것보다는 사람이 이해할 수 있는 이름을 이용하는 것이 바람직할 것이다. 이름 해석법은 TCP/IP상에서 수행되며, 크게 2가지로 나누어진다. 컴퓨터는 논리적 명칭 해석에 호스트명 해석법, NETBIOS 해석법 중 하나를 사용하여 IP주소로 변환하는데 사용한다[8].

1.1 호스트(HOST)명 해석법

호스트명 해석법은 인터넷의 초기부터 계속 사용되어왔다. 원래 모든 정보는 호스트파일에 집중적으로 기록되었다. 인터넷상의 임의의 새 호스트들은 그들의 호스트명을 등록시키고, 네트워크 정보센터(NIC)에 의해 중앙 호스트파일의 IP주소가 유지되었다. 그리고 인터넷상의 모든 호스트들은 집중된 호스트파일을 서로 공유하였다. 그러나 인터넷이 증가하면서 다음과 같은 집중된 호스트파일로 인해 많은 문제가 발생했다.

- 인터넷의 확대로 인하여 갱신이 매일 기본적으로 일어난다.
- 호스트파일을 관리한 스탠포드 연구소(Stanford Research Institute)의 네트워크는 인터넷에서 많은 장애가 되었다.
- 호스트명은 명칭 공간의 단조로운 성질 때문에 인터넷 어디에서든 복제될 수 없었다.
- 명칭 갱신들은 총괄적으로 인터넷에 보이는데 많은 시간이 걸렸다.

위와 같은 문제로 인해 제안된 해법들은 "계층적 이름해석법이라는 접근방법과 또 다른 중요한 제안은 데이터베이스가 집중적이라기보다 오히려 특성상 분산적이라는 사실이었다. 이 방법으로, 각 조직은 그들 자체의 호스트 명들을 유지, 관리할 수 있었다.

도메인 네임 스페이스는 인터넷에서 네임 스페이스를 구성하는 모든 도메인들을 나타내는 트리 구조이다. 루트 도메인은 트리(Tree)의 꼭대기에 있다. 루트 도메인은 실제 텍스트 레이블을 가지고 있지 않다. 그것은 마침표(.)를 사용하여 표현된다. 루트도메인 아래는 최상위 계층 도메인들이다. 최상위 계층 도메인들의 두 가지 성격으로 구분된다. 첫 번째 분류에서, 상위계층 도메인들은 업무의 타입들을 표현한다. 두 번째 분류는 조직(국가)이 위치한 지역을 나타내는 두 자리 코드이다. 일반적인 최상위 계층 도메인들은 도메인 네임 스페이스를 확장하는데 필요하기 때문에 추가되었다. 호스트들과 서브도메인들은 포함하는 제 2계층 도메인들은 최상위 계층 도메인들 아래에 존재한다. [그림 2]는 최상위 계층 도메인에 제 2계층 도메인으로 등록된 조직의 예이다.

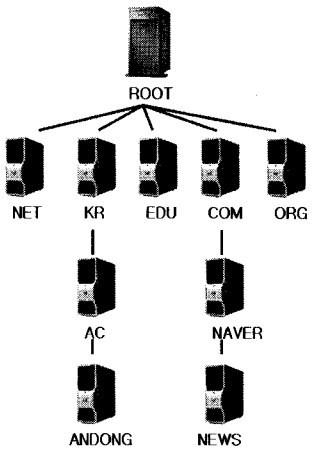


그림 2. DNS 계층구조

KR은 지역 도메인 내에서 일반적인 분류 기법을 나타낸다. 이 경우에 그것들은 "한국"을 의미하고, 다시 "CO", "AC" 등 다시 업무의 성격을 의미한다. 이는 국가의 최상위 계층 도메인 내에서 완벽한 업무적인 분류로 지정된다. CO와 AC는 KR인 제 2계층 도메인 아래의 서브도메인들이다. 만약 NAVER에서 웹 서버의 웹 페이지를 보고 싶다면, 웹 브라우저에서 NAVER.COM

사이트에 대한 URL(Uniform Resource Locator)을 입력한다. 풀 도메인 네임 경로를 포함하는 호스트명은 FQDN(Fully Qualified Domain Name)이라 불린다. FQDN은 응용 프로그램으로 하여금 호스트에 대한 절대 경로가 URL에 기술됨을 알 수 있다.

2. DNS 동작 및 구성요소

2.1 호스트명 솔루션 프로세스

컴퓨터가 호스트명을 IP주소로 변환하는 처리과정은 다음과 같다.

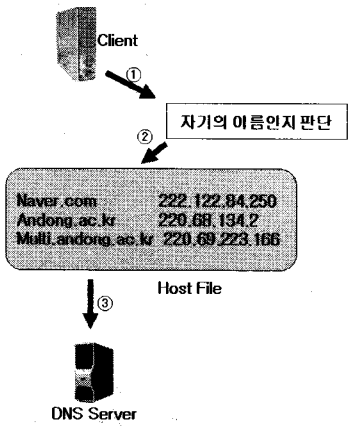


그림 3. IP해석의 기본절차

① 바꾸려는 도메인이 작업 중인 자기 자신의 호스트인지 판단

② 물으려 하는 도메인이 HOSTS 파일에 있는지 판단

③ 원격 DNS서버가 이 도메인에 대하여 정보를 가지고 있는지 판단

만약 이러한 해석 방법들의 그 어떤 것도 목표 호스트명에 대한 IP주소를 찾지 못하면, 응용 프로그램은 호스트 명을 찾을 수 없음을 의미하는 에러 메시지를 되돌린다.

3. DNS의 보안 취약점 및 대처방안

AT&T의 Bell 연구소의 벨로빈(Steven M. Bellovin)은 DNS보안 취약성에 대한 연구를 통해 DNS공격이 이름 기반의 사용자 인증 문제와 도메인 네임 시스템에 대한 접근문제, 변경된 정보에 대한 진위 여부 검증불가 등과 같은 DNS 시스템이 갖는 근본적인 취약성을 지적했다[9]. IETF에서는 DNS의 보안적인 취약점을 몇 가지 관점에서 분류하고 있다.

3.1 서비스 관점

서비스관점은 DNS의 취약성은 공격자에 의해 위/변조된 정보를 서버가 클라이언트에게 제공함으로서 공격자가 의도한 목적지로 사용자를 유인하거나, 서비스 거부 공격(Denial of Service)을 가능하게 한다. 질의에 대한 널(null) 응답, 잘못된 질의응답(redirect, inject), 서비스거부 등이 여기에 속한다.

3.2 공격형태 관점

서비스관점에서는 패킷 가로채기를 변형시킨 다양한 위/변조 공격이 존재한다. 이를 통해 DNS는 공격자에 의해 변조된 잘못된 응답을 클라이언트에게 보내는 취약점이 있다.

3.3 공격대상 관점

공격대상 관점에서 살펴 보면 DNS 공격자는 호스트 네임 스푸핑(Host Name Spoofing), DNS 스푸핑(DNS Spoofing) 등을 통해서 DNS 클라이언트와 네임서버의 정보에 대한 위/변조 공격을 할 수 있다.

3.4 소프트웨어 취약성 관점

소프트웨어 취약성에 대한 공격은 일반적인 시스템과 네트워크에 알려진 공격으로서 DNS에만 한정된 보안 취약성이라고 할 수는 없다. 하지만, 악의의 코드를 실행하여 공격자가 의도한 바를 수행하거나 DNS서버에 대한 오버플로 공격을 통해 서비스 자체를 유용하지 않게 할 수도 있다.

3.5 DNSSEC(DNS Security Extensions)

DNS의 보안 취약점에 대한 해결책으로 제시되고 현재까지 지속적으로 연구 개발되고 있는 개념이

DNSSEC(DNS Security Extensions)이다.

DNSSEC는 DNS에서 신뢰할 수 없는 DNS요청에 대한 응답을 디지털 서명하여 악의적인 목적을 갖는 사용자가 데이터를 악용하지 못하게 하는 데 목적이 있다. IETF내에서 DNS연구와 발전을 주도하는 dnsext (DNS Extensions)와 dnsop(DNS Operations)에서는 DNS 보안 기술에 대한 세부적인 프로토콜 기준을 제시하고 표준화작업을 하고 있다.

그러나 DNSSEC가 적용되면 한 메시지의 크기가 현재의 6배 정도가 증가하게 되며 헤더의 크기 증가와 통신 설정에 대한 비용적인 측면이 고려되어야만 한다. 또한, DNSSEC에서 사용되어지는 인증키에 대한 키관리를 위해 키를 분배해야하는 엄청난 작업을 해야만 한다는 것이다. 게다가 현재 인터넷의 구성상 전 세계를 망라할 수 있는 공개키 기반 시스템이 존재하지 않고, 관리적인 측면에서도 많은 어려움을 가져오게 될 것이다[10].

4. DNS 계층구조의 장,단점 및 피해사례

인터넷은 최상위에 루트 도메인(“.”으로 표시)이 있고, 루트 도메인에 여러 개의 TLD(Top Level Domain)으로 구성되어 있다. 예를 들어 .kr, .jp 와 같은 국가 기반의 도메인(CCTLD, Country Code Based Top Level Domain)과.com, .net 등 국가를 초월한 도메인(GTLD, Global TLD)이 있다. 이 최상위 레벨(top level) 도메인 밑에 하위 도메인이 생기고 그 계층은 정해진 바는 없다. 각 도메인에는 그 도메인에 관한 이름 정보를 보관하고, 실시간의 질의에 응답하는 서버가 있다. 이 서버들을 DNS(Domain Name Service) 서버라고 하고, 우리는 이 DNS서버들의 서비스를 이용하여 도메인 네임을 인터넷주소로 변경할 수 있는 것이다. 앞서 살펴본 것과 같이 현재의 DNS계층적 구조를 가지고 서로 협력하는 형태로 구성되어 있다. 이와 같이 계층 구조는 자연히 최상위 DNS가 많은 도메인 해석요청을 감당해야 한다는 단점이 존재한다.

핵심 Root DNS는 DNS 프로토콜의 응답 패킷인 UDP(User Datagram Protocol) 내부에 수용된 서버 수의 최대치가 13이기 때문에 전 세계에 13개(미국10개, 유럽 2개, 일본 1개)가 가동 중이며 인터넷의 핵심이라 할 수 있다. 우리나라 kr네임서버는 국내에 한국통신(KT), LG데이콤(LG Dacom), 한국과학기술정보연구원(KISTI), 하나로텔레콤(Hanaro Telecom), 한국인터넷진흥원(NIDA), 한국정보사회진흥원(NIA) 등 6개 기관이 서울에 4개 사이트, 안양에 대전에 각 1개의 사이트를 운영중이며, 국외에는 미국에 2개, 중국에 1개 그리고 독일에 1개의 사이트를 운영하고 있다[11]. 하지만 반대로 생각하면 핵심이 되는 13개의 Root DNS와 같은 상위 DNS가 장애를 일으키면, 인터넷이 마비되는 사이버대란 같은 사태가 올 것이다. 실제로 국내에서 2003년 1월 25일 국내에서 초유의 인터넷 대란이 발생했다. 두 개의 백본망을 쓰는 국내의 경우 KT의 "혜화전화국"과 "구로전화국" 두 곳에서 DNS가 작동하고 있는데, 이날 문제가 된 DNS 서버는 혜화 전화국에 위치한 것이다. 이 두 곳에서 처리하는 접속신호는 국내 인터넷 접속의 80% 정도를 차지할 정도다. 이날 인터넷 마비 사태는 혜화 전화국의 DNS서버가 다운되자, 구로 전화국의 DNS서버로 접속신호가 몰려 이마저도 과부하가 걸려 발생한 것으로 파악되고 있다. 주말에 불과 9시간 동안 벌어졌던 사상초유의 인터넷 불통 사태는 국내 사이버 세상을 순식간에 "암흑"으로 만들면서, 사회적으로 엄청난 충격을 안겨다 주었다. 웹 사이트가 차단되고, 인터넷 예약이 이뤄지지 않고, e-mail과 인터넷뱅킹이 중단되고 전자상거래가 마비되면서 큰 혼란이 발생한 것이다. 물론 금액으로 환산하기 힘들만큼 경제적 손실이 있었다.

이러한 인터넷대란의 원인은 슬래머 웜 바이러스의 공격으로 국내 상위 DNS가 마비되면서 IP주소 해석을 할 수 없어서 각 클라이언트가 웹서버를 찾지 못해 발생한 문제이다. 실제로 대부분의 웹 서버는 정상적으로 가동되었지만, 클라이언트가 IP주소 해석을 받지 못해서 웹서버에 접속할 수 없었다. 이러한 작은 바이러스의 공격으로도 거대한 인터넷 망이 접속마비로 이어질 수 있다는 점은 우리에게 시사하는 바가 매우 크다. 더 큰 문제는 이러한 DNS서버 취약점을 여전히 완벽하게 대응할 수 없다는 것이다. 그리고 우리는 여전히 이런 위험에서 자유롭지 못한 것이 현실이다. 이 문제는 단

지 보안만 강화한다고 완전히 해결될 수 있는 문제는 아니다. 결국 다른 대안이 있어야 할 것이다[12].

다음 장에서 수직적 계층구조를 가진 지금의 DNS구조의 장점을 그대로 따르면서 보완책으로 수평적, 독립 DNS구조를 추가한 개선안에 대해서 설명한다.

III. 제안하는 DNS장애 발생시 대처방안

현재 사용하고 있는 계층적 DNS의 구조를 간단히 다시 살펴보면 아래 [그림 4]처럼 계층구조를 이용하여 인터넷주소를 해석한다. 그러나 루트 DNS나 상위 DNS가 장애로 인해서 정상적인 역할을 수행할 수 없다면 그 하위에 있는 DNS가 정상적으로 동작한다고 해도 연결고리가 끊어지기 때문에 자기 역할을 정상적으로 수행할 수 없다. 물론 각 DNS영역에서 보조 DNS 서버를 운영한다고 해도 여전히 위험성은 존재한다. 현재의 계층구조는 정보를 분산할 수 있는 구조이고, 트리구조에서 빠른 검색이 가능하다는 장점 때문에 지금의 계층 구조를 포기할 수 없다.

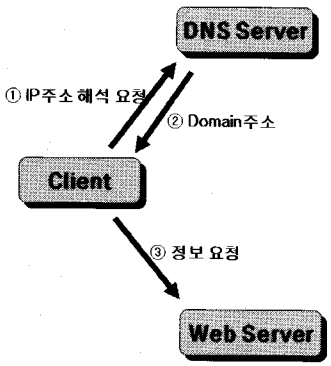


그림 4. DNS 주소 해석 구조

이와 같은 문제의 개선안으로 지금의 계층구조의 장점을 유지하면서, 보완으로 수평적, 독립 DNS구조를 추가한 것이다. 평소에는 지금의 계층구조와 독립(로컬) DNS구조를 병행 운영하고, DNS장애 발생 시 독립구조와 수평구조를 병행하여 운영한다. 개선방안의 장점으로는 어떠한 장애에도 DNS의 운영이 가능하다는 점을 들 수 있다. 그리고 자신만의 로컬 DNS의 운영으로 상위 DNS의 부하를 최소화 할 수 있다는 것이고, 자신의 DNS정보를 이용할 때는 인터넷주소 해석과정이 생략되므로 빠른 접속이 가능하다는 것이다. [그림 5]는 지금의 계층구조에 자신만의 로컬 DNS와 각각의 로컬 DNS를 수평적 구조로 결합한 것이다.

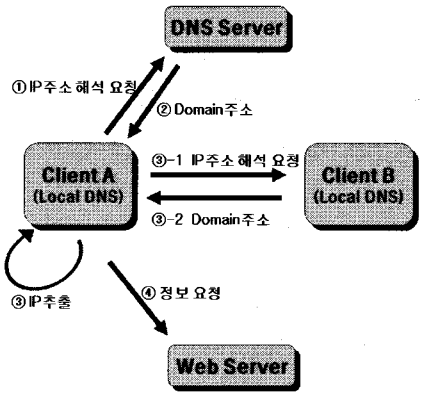


그림 5. 개선안 구조(계층 + 수평 + 독립 DNS 구조)

개선안 구성과 기본적인 동작원리는 다음과 같다. 먼저 응용프로그램은 접속하고자 하는 웹 서버의 주소를 자신의 로컬 DNS에서 먼저 조회를 시도한다. 이러한 기능은 현재 NT나 리눅스 시스템에서 사용 중인 호스트파일을 조회하는 것과 같은 원리이다. 차이점은 지금 사용 중인 호스트파일은 사용자가 일일이 수작업으로 자료를 갱신해 주어야 하기 때문에 사용자가 실제로 사용하기는 어려움이 많은 것이 현실이다. 본 논문에서 제안하는 방식은 호스트파일을 수작업으로 일일이 갱신하는 것이 아니라 응용 프로그램의 요청을 받아서 동적으로 호스트 파일이 자동적으로 구성되는 것이다. 정보가 중복 저장되지 않고, 실시간 갱신도 이루어진다는 것이다. 이러한 동적 기능을 가짐으로써 사용자는 전혀 의식할 필요 없이 평소처럼 인터넷을 사용하기만 하면 된다. 그리고 자신의 DNS에서 원하는 IP주소를 찾지 못하면, 주변에 있는 다른 로컬 DNS에게 도메인해석을 요청한다. 만약 주변에 있는 다른 DNS도 정보를 가지고 있지 않다면, 또 다른 로컬 DNS의 인터넷 주소를 통보한다. 이와 같은 과정을 반복해서 원하는 인터넷주소

를 찾게 된다.

IV. 시스템 설계 및 구현

1. 독립(로컬) DNS의 구조

본 논문에서 제안하는 독립(로컬) DNS는 [그림 6]처럼 응용 프로그램과 같이 자신의 컴퓨터에서 구동된다. 구성요소는 캐시메모리와 파일시스템 또는 데이터베이스를 이용하여 DNS자료를 관리한다. 동작원리는 다음과 같다.

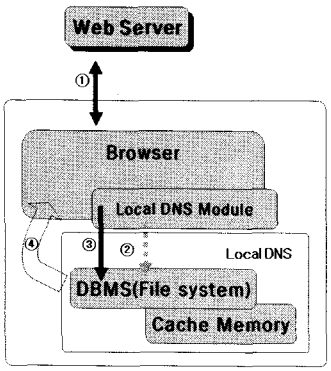


그림 6. 독립(로컬)DNS 내부 구조(DNS 정상 운영 시)

① DNS가 정상일 때는 웹 브라우저 주소 창에 입력된 URL정보로 웹서버에 정상적으로 접속이 되고, 로컬 DNS 정보에 해당 도메인의 IP를 저장한다.
② 로컬 DNS 모듈은 URL을 파싱(Parsing)하여 서버 도메인 주소만 추출하여 IP와 함께 저장한다.
③ 만약, DNS가 정상적으로 작동하지 않아 웹서버의 접속이 정상적으로 되지 않으면 입력된 URL을 파싱하여 도메인 추출 후에 DBMS와 캐시메모리에 있는 해당 IP를 검색한다.
④ 로컬 DNS에서 검색된 IP값을 웹 브라우저로 보낸다.

상위 DNS가 정상적으로 운용 중일 경우는 ①에서 검색된 IP를 로컬DNS의 호스트파일에 저장 갱신하게 된다.

[그림 7]과 같은 자료처리 과정을 통하여 사용자는 수작업 없이 로컬 호스트파일을 동적으로 구성하게 되므로 편리하게 사용할 수 있다. 이렇게 수집된 DNS정보는 사용자에게는 인터넷 접속속도를 증가시키고, 상위 DNS의 부하를 줄일 수 있다는 장점이 있다. 그리고 가장 큰 장점으로 DNS장애 발생 시 수집된 정보를 이용하여 상위 DNS의 도움 없이, 제한적이지만 과거의 방문한 웹 서버에 접속할 수 있다.

```
If isLiveDNS(Domain) Then
    ip = checkICMP(Domain)
    if isNotDomainInDBMS then
        saveDomain(Domain, ip)
    else
        updateDomain(Domain, ip)
    end if
else
    ip = searchDBMSandCache(Domain)
    if IPisNotExistinLocalDNS then
        searchOtherClientDNS()
    end if
    pushIPtoBrowser(Domain, ip, BrowserName)
end If
```

그림 7. 독립(로컬) DNS 모듈 알고리즘

2. 수평적 구조의 DNS의 설계 및 구현

수평적 구조의 DNS는 독립된 로컬 DNS의 단점을 보완하는 기능이다. 앞에서 살펴본 것 같이 로컬 DNS의 가장 큰 단점은 모든 DNS정보를 가질 수 없다는 것이다. 그 이유는 항상 갱신되는 정보를 바로 파악할 수 없고, 많은 양의 정보를 로컬 DNS가 관리한다는 것은 부담스럽기 때문이다. 그래서 자신에게 없는 DNS정보를 다른 컴퓨터의 로컬 DNS에 질의를 통하여 상호 보완하는 것이다.

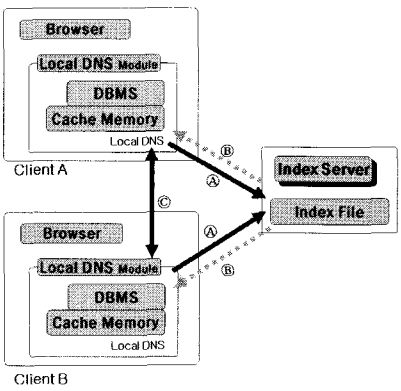


그림 8. 수평적 구조의 DNS 연결 구조

① 컴퓨터 A의 로컬 DNS가 실행되면 먼저 A의 IP와 호스트파일에 저장된 IP주소의 개수를 인덱스 서버에 통보한다. 인덱스서버는 각 컴퓨터가 보내온 자료를 호스트파일을 이용하여 유사시 대비하여 보관한다. 그리고 컴퓨터 A에게 어떤(정보가 많은) 컴퓨터 B의 IP주소를 (링크주소) 알려준다. 컴퓨터 A는 이렇게 인덱스 서버에게 받은 B의 IP주소를 유사시 수평연결을 위해서 변수 형태로 메모리에 보관한다. (Ⓐ)

② 웹 브라우저 주소 창에 입력된 URL정보를 로컬 DNS로 보낸다.

③ 로컬 DNS는 URL을 파싱하여, 디렉토리 주소를 제외한 서버 도메인 주소만을 추출한다. 추출된 도메인 주소를 배열 형태로 저장된 캐시 메모리를 기준으로 순차검색을 시도한다. 만약 해당 IP주소를 찾지 못하면 ④로 가고, 찾으면 IP주소를 웹 브라우저로 보낸다.

④ ③번 자신의 캐시 메모리에서 IP주소를 찾지 못하면 로컬 DNS에 저장된 호스트 파일을 검색한다.

⑤ 해당 IP주소를 찾으면 IP주소를 웹 브라우저로 보낸다.

⑥ 만약 A의 호스트 파일에서 해당 IP주소를 찾지 못하면, 인덱스 서버 캐쉬 메모리에 질의를 요청해 보고, 없으면 ①에서 구한 링크주소(B의 IP주소)를 이용하여 B의 로컬 DNS에게 도메인 해석을 요청한다. B는 A의 요청을 자신의 호스트 파일에서 검색 후 IP주소를 찾으면, 이를 A에게 보낸다. (Ⓑ)

⑦ 만약 B도 IP정보가 없다면 B가 가지고 있는 링크주소를 A에게 보내준다. A는 다시 B가 보내온 링크주소를 이용하여 다른 컴퓨터에 다시 질의를 요청한다. 이후 IP주소를 찾을 때 까지 반복한다. 그 결과 A는 IP주소를 획득한 후, 그 결과를 인덱스 서버의 캐시 메모리에 통보하고, 인덱스서버는 다른 클라이언트 요청이 있을 때 이를 서비스할 수 있다. 최적의 DNS정보를 찾기 위한 알고리즘은 다음 [그림 9]와 같다. (Ⓒ)

이와 같은 반복적인 자료처리 과정을 통해서 다른 컴퓨터에 있는 DNS정보를 상호 참조, 공유할 수 있도록 하였다.

```
If isLiveDNS(Domain) Then
    call LocalDNSModule
    putLocalInfotoIndexServer(LocalDNSinfo)
else
    if IPisNotExistinLocalDNS then
        loop
            OtherInfo = getDNSInfofromIndexServer
            ArrayIP = OtherInfo.IPList
            ArrayDomain = OtherInfo.DomainList
            Search(Domain, ArrayDomain)
            if isExistDomain then
                ip = ArrayIP(Search)
                Exit loop
            end if
        end loop
    end if
    pushIPtoBrowser(Domain, ip, BrowserName)
end If
```

그림 9. 최적의 DNS를 유지하기 위한 알고리즘

3. 제안 DNS 구조 분석

본 논문에서 제안한 DNS구조는 평소 계층구조와 독립 DNS구조를 병행 운영하면서, 상위 DNS장애 발생 시 독립구조와 수평구조를 통해 어떠한 장애에도 DNS가 운영이 가능하고 자기만의 로컬(독립) DNS운영으로 상위 DNS의 부하를 최소화할 수 있다. 그러나 DNS가 정상 동작할 때 평소 자신이 많이 쓰는 도메인 정보가 동적으로 관리됨으로써, 수집되는 정보량이 굉장히 많아질 수 있어 그것들을 효율적으로 관리할 필요가 있다.

[그림 10]은 독립DNS 정보를 수집하는 과정을 시뮬레이션 한 것이다. DNS가 정상적으로 운영 중일 때는 해당 IP를 로컬 데이터베이스에 저장하게 된다. (①②)

또한, DNS 장애가 발생해서 주소해석의 문제가 생기면 로컬 데이터베이스에 저장된 IP주소를 이용해서 접속하게 된다. 장애가 발생하여 수평구조 사용 시 접속시간이 길어질 수 있고 최악의 경우 원하는 정보를 찾지 못할 수 있다. 수집가능 한 형태의 시뮬레이션을 통해서 수집된 정보량의 관리는 최근 컴퓨터 성능의 급속한 증대로 큰 문제가 되지 않는다. 또한 접속시간이 길어지는 문제점은 평소에는 발생하지 않고 상위 DNS장애 발생시에만 문제가 되기 때문에 제안된 DNS구조 사용 시 대부분의 인터넷 서비스를 중단 없이 사용할

수 있는 장점에 비해 무시될 수 있었다. 이러한 특징으로 인해 인터넷 마비에 따른 경제적 손실을 최소화할 수 있으므로 본 논문에서는 제안한 DNS구조가 기존 DNS체계보다 안정적인 구조를 가진다는 것을 확인할 수 있다. 또한 본 논문에서 제안한 방안은 기존의 DNS 망을 그대로 두고 진행될 수 있기 때문에 적은 비용으로도 구축할 수 있다.

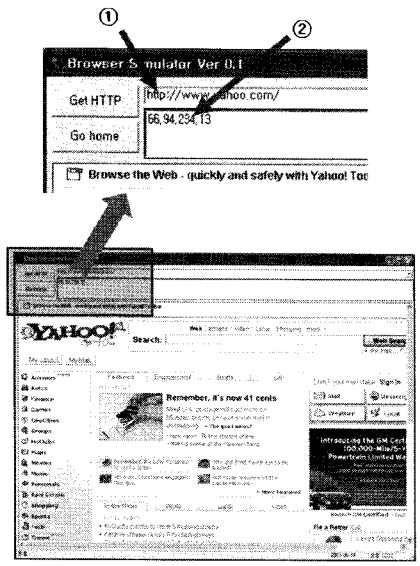


그림 10. 로컬(독립) DNS 시뮬레이터 동작 화면

V. 결 론

지금의 계층적 DNS가 초기 인터넷인 ARPANET시절부터 존재했던 것은 아니다. 초기에는 네트워크에 연결된 호스트의 수가 많지 않았기 때문에 텍스트파일 형태의 호스트파일에 호스트이름과 이에 대응되는 IP주소를 입력하는 것으로도 충분하였다. 그러나 네트워크가 급격히 확장되면서, 모든 호스트가 NIC에 접속하여 호스트파일을 받는다는 것이 심각한 문제가 되었고, NIC에서 호스트파일을 관리한다는 것이 불가능한 상황까지 이르게 되었다. 이러한 문제점을 보완한 새로운 이름 해결 방법이 바로 계층구조의 DNS이다. 지금의 DNS는 계층적인 네임스페이스를 이용하여 네트워크 확장에 대비한 확장성을 제공하며, 분산 관리를 지원할 수 있기 때문에 상당히 효율적인 방법이다. 그러나 지금의 DNS의 구조는 계층구조의 특성상 연결구조에서 취약하다는 문제점, 개방된 환경, 악의적인 공격으로부터 노출되어 있다는 것이다. 물론 보조 DNS서버를 운영한다고 해도 여전히 위험성은 존재한다. 지금의 계층구조는 정보를 분산할 수 있는 구조이고, 빠른 검색이 가능하고, 효율적인 자료관리가 가능하기 때문에 지금의 계층구조를 포기할 수는 없지만 문제점을 보완하기 위한 시도가 필요하다.

본 논문에서 제안된 방안은 지금의 계층구조를 그대로 따르면서 수평적 독립 DNS구조를 추가하여 평소에는 현재의 계층구조와 로컬 DNS구조를 병행 운영하고, 상위 DNS장애발생시 독립구조와 수평구조를 병행하여 운영한다. DNSSEC의 적용에 대한 엄청난 비용, 세계적인 키 분배 문제 등에 대한 명확한 해결책이 없는 현재로서는 본 논문에서 제안된 방안이 DNS의 보안 취약점을 일부 보완하면서 안정적이고 신뢰성 있는 인터넷사용을 보장할 것이라고 기대된다.

따라서 클라이언트 측면에서는 상위 DNS의 문제가 발생하더라도 클라이언트 자체에 저장되어진 정보를 이용하여 인터넷을 동일하게 사용 가능하게 되는 것이다. 결과적으로 모든 DNS의 운영을 효율적으로 할 수 있으며, 클라이언트는 좀 더 안정적인 인터넷의 접근이 가능하게 된다. 또한 클라이언트 자체 DNS 정보를 사용함으로써 인터넷주소의 해석과정이 생략되어 빠른 접속이 가능할 뿐 아니라, 악의적인 DNS공격에도 능동적으로 대응할 수 있는 시스템에 활용할 수 있을 것으로 기대된다.

참 고 문 헌

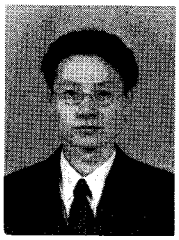
[1] H. Ogawa and Y. Goro, *Web 2.0 Book*, Impress Japan Corporation, 2006.

[2] Unlyess Black, "Computer Networks-Protocol ,Standards and Interfaces," Prentice Hall, pp. 261-285, 1993.

[3] R. Lemos, "Net attack-how it was squashed," CNET News.com, Oct. 2002.
[4] B. Komar, "TCP/IP Network Administration," pp.202-229, 1999.
[5] http://www.etnews.co.kr/news/detail.html?id=2006022400 25
[6] http://news.naver.com/news/read.php?mode=LSD&office_id=030&article_id=0000057637&section_id=105&menu_id=105
[7] B. A. Forouzan, "TCP/IP Protocol Suite," McGraw Hill, pp.152-177, 2001.
[8] U. D. Black, "Computer Networks," Prentice Hall, pp.261-286, 1993.
[9] 유신근, 이현우, *DNS 안전운용가이드*, CNRT/CC-kr, June 2002.
[10] 김학주, 윤민우, 임형진, 송관호, 정태명, "도메인 네임시스템의 취약점 분석과 보안 확장(DNSSEC)", 한국통신학회지, 제20권, 제7호, pp.14-27, 2003.
[11] http://domain.nida.or.kr/information/namecondition.jsp
[12] Chuck Easttom, *Computer security fundamentals*, Prentice Hall, 2006.

저 자 소 개

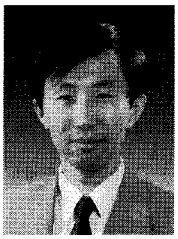
임 양 원(Yang-Won Lim) 정회원



- 1992년 : 충주대학교 컴퓨터공학과(공학사)
- 2001년 : 안동대학교 컴퓨터공학과(공학석사)
- 2004년 : 안동대학교 멀티미디어공학전공 박사과정

<관심분야> : 멀티미디어, 에이전트, 웹프로그래밍

임 한 규(Hankyu Lim) 종신회원



- 1981년 : 경북대학교 전자계산기공학전공(공학사)
- 1984년 : 연세대학교 전산전공(공학석사)
- 1997년 : 성균관대학교 컴퓨터공학전공(공학 박사)
- 1998년 3월 ~ 현재 : 안동대학교 멀티미디어공학전공 교수

<관심분야> : 멀티미디어, 웹응용, 자연언어처리