웹 서버 클러스터 환경에서의 보안세션 고속화 기술

진 승 의† · 김 태 일† · 이 형 호††

요 약

인터넷을 이용한 전자상거래(e-commerce)가 확산되면서 중소규모의 다중 서버를 클러스터화하여 클라이언트의 요청에 대한 응답을 수행하는 네트웍 서비스에 대한 요구가 증대하고 있다. 이러한 웹 서버 클러스터 환경에서 다중의 서버들이 원활하게 상호연동을 수행하여 클라이언트의 요청을 처리하기 위해서는 TCP 정보나 요청되는 대상의 컨텐트(content) 정보를 이용하여 라우팅을 수행하는 라우터의 구현이 필수적이다. 이러한 TCP 라우팅이나 컨텐트 라우팅의 구현을 전적으로 소프트웨어에 의존하는 경우 전송속도의 저하가 발생할 우려가 있다. 이는 기존의 TCP 커넥션 설정과 이 TCP 커넥션을 이용한 데이터 전송이 라우팅을 고려하지 않은 가상의 점대점(point-to-point) 연결구조를 갖기 때문이다. 따라서, 본 논문에서는 보안세션 라우팅을 고속화시키기 위하여 TCP 커넥션 고속화를 기반으로 보안세션의 재사용성을 향상시키는 웹 서버 클러스터 환경에 최적화된 디스패칭(dispatching) 기법을 제시하고자 한다.

High Speed Secure Session Technology for Web Server Cluster

Seung-Eui Jin† · Tae-Il Kim† · Hyeong-Ho Lee††

ABSTRACT

The Web-based e-commerce causes the exponential growth of the Internet users and makes the security on the Internet important. To provide scalability for these increased bandwidth needs, it has been proposed that a pool of back-end Web servers are clustered and load-balanced by a dispatcher. Guaranteeing the security on an open public network, such as the Internet, needs time-consuming handshake protocol for authentication and encryption. Therefore, to avoid unnecessary bandwidth waste, the previously negotiated secure session should be reused. However, in the server cluster environment, load balancing mechanism conflicts with the session reuse efficiency. In this paper, we make close investigation into the TCP Splicing and content-aware dispatching technique and propose the method to improve the secure session reuse efficiency in cluster-based Web server systems.

키워드 : 웹서버 클러스터(Web Server cluster), 전자상거래(E-commerce), 가상사설망(Virtual Private Network), 보안 프로토콜(Security Protocol)

1. 서 론

전자상거래가 활성화됨에 따라 웹(Web)을 이용한 데이터 서비스의 수요가 급증하게 되었다. 특히, 전자상거래에서 트랜잭션(transaction)되는 데이터에 대한 보안의 필요성이 강조되면서 이를 위한 암호화(encryption)와 인증과정(authentication)을 수용하는 방향으로 네트웍구조가 진화하고 있다. 이렇게, 특수 목적을 수행하는 기능성이 네트웍에 추가됨에 따라 네트웍을 구성하는 라우터(router)들과 스위치(switch) 시스템들이 겪게 되는 부하는 점점 증가하고 있다. 이로 인하여 전송속도에 대한 고려가 없이는 전체적인 네트웍의 전송성능은 자연히 저하되게 될 것이다. 그러므로, 현재 각광을 받고 있는 파장분할다중(WDM : Wavelength Division Multiplexing) 광전송 기술개발과 같이 네트웍 코어(core)의 전송속도를 극대화하려는 노력도 네트웍 서비스의 양 종단에 위치하는 클라이언트(client)와 서버(server)간의 커넥션이 특수 목적의 기능을 수행하기 위하여 전송속도의 저하를 감수해야 한다면 전체적인 네트웍 성능 측면에서 의도하였던 전송 속도의 극대화는 기대할 수 없게 된다.

현재의 많은 네트웍 서비스들이 TCP/IP를 바탕으로 그 위에서 구현되고 있다[1]. 기존의 라우팅(routing) 과정은 longest prefix match를 수행하기 위하여 이를 소프트웨어적으로 처리함으로써 높은 전송속도를 내기가 어려웠으나, 최근에 들어 고속 전송을 수행하기 위하여 라우팅 과정을

† 정 회 원 : 한국전자통신연구원 라우터기술연구부
†† 종신회원 : 한국전자통신연구원 라우터기술연구부
논문접수 : 2001년 4월 16일, 심사완료 : 2001년 7월 3일

단순화하고 하드웨어적으로 라우팅이 가능하게 하는 포워딩 엔진(forwarding engine)[2, 3]이 개발되어 IP라우팅의 고속화가 이루어지게 되었다. 그러나, TCP의 경우는 트랜스포트 계층(transport layer)에 존재하여 응용계층과 밀접하게 연동함에 따라 그 자체가 소프트웨어적인 성향이 강하기 때문에 고속화를 위한 기법을 적용하기가 쉽지가 않은 특성을 가지고 있다.

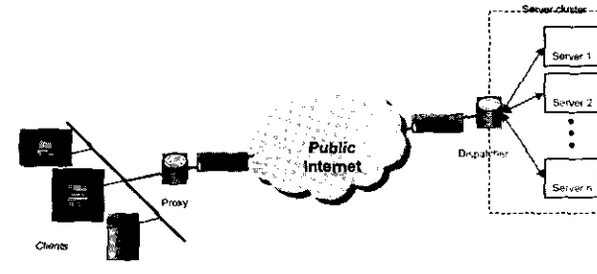
특히, 다음 장에서 소개할 웹 서버 클러스터 환경과 같은 경우는 중소규모의 다중 서버들이 하나의 그룹을 형성하고 그룹내의 서버들이 동시에 연관된 컨텐트를 제공하는 서비스를 수행하고 있다. 이러한 환경에서 네트웍 에지(edge)에 존재하는 라우터나 스위치가 IP 라우팅뿐만 아니라 TCP 라우팅도 수행하여야 할 필요가 있게 된다. 본론에서 자세히 설명하겠지만, 좀더 효율적인 웹 서버 클러스터의 운영을 위해서는 TCP 라우팅과 더불어 부분적인 컨텐트 라우팅의 기능도 추가되어야 한다. 여기서 컨텐트 라우팅이란 HTTP와 같은 응용서비스의 URL(uniform resource locator) 정보 나 클라이언트의 쿠키(cookie) 정보를 참조하여 이에 맞게 라우팅을 하거나, 보안이나 QoS(Quality-of-Service) 서비스를 위하여 형성된 특수세션의 재사용성을 높이기 위하여 세션 식별자(session identifier)에 의한 라우팅을 수행하는 것을 의미한다. 이러한 TCP나 컨텐트 라우팅을 위해서는 프로토콜 스택(protocol stack)에서 트랜스포트계층(L4 layer)과 응용계층(L5 layer)의 상위계층까지 취급하여야 하기 때문에 매 패킷을 전송할 때마다 라우터나 스위치에 탑재된 주프로세서의 운영체제에 존재하는 프로토콜 스택의 최상위까지 패킷이 전달되어 소프트웨어적으로 처리되어야 한다. 그러나, 이러한 과정은 매 패킷을 처리할 때마다 많은 양의 프로세스를 생성하고 데이터의 복사가 빈번하게 이루어지기 때문에 고속의 네트웍에는 바람직하지 않게 전송속도 저하를 초래하게 된다. 따라서, TCP나 컨텐트 라우팅 과정의 전체를 하드웨어화할 수 없지만 부분적으로 하드웨어적인 처리기법을 도입함으로써 라우팅을 통한 전송속도 저하를 막을 수 있는 방안을 제공하는 것이 필수적이게 되었다.

본 논문에서는 앞에서 기술한 TCP 라우팅과 컨텐트 라우팅을 웹 서버 클러스터 환경에서 구현하고자 할 때 전송성능의 저하를 방지하기 위하여 개발된 TCP 고속화 기법에 대하여 알아보고, 이를 이용하여 전자상거래에 핵심이 되는 보안세션(secure session)을 고속화하는 기법을 제안하고자 한다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 전자상거래를 위하여 네트웍 양 종단에 필수적으로 요구되는 웹 서버 클러스터 구조에 대하여 간단히 소개한다. 3장과 4장에서는 웹 서버 클러스터 환경에서 고속으로 세션을 설정하고 전송을 수행하기 위한 TCP 고속화 기법으로 TCP Splicing과 컨텐트-인지형 디스패칭(content-aware dispatching) 기술에 대하여 설명한다. 그리고, 5장에서는 TCP Splicing과 컨텐트-인지형 디스패칭 기술을 활용하여 고속의 보안세션을 설정하고 세션 데이터를 포워딩하는 기법을 제안하고 6장에서 향후연구 방향에 대하여 논의하고 결론을 맺는다.

2. 웹 서버 클러스터 구조

중소규모의 다중 서버들로 구성된 웹 서버 클러스터를 수용한 네트웍 구조를 (그림 1)에 나타내었다. 그림에서 보는 것처럼 일반적으로 클라이언트들로 구성된 네트웍 도메인(domain)은 프록시에 의하여 관리되고, 인터넷과 같은 공중망(public network)을 통하여 접속되는 서버들도 디스패처(dispatcher)라고 명명되는 프록시에 의하여 관리되는 구조를 갖게 된다.



(그림 1) 웹 서버 클러스터를 수용한 네트웍 구조

웹 서버를 이용한 전자상거래에 대한 관심과 수요가 증가하면서 이러한 웹 서버 클러스터 구조를 수용한 네트웍의 수효가 증가하고 있다. 따라서, 확장성(scalability)을 보장하는 웹 서버 클러스터 네트웍 환경을 구축하는 것은 중요한 기술적 이슈가 되고 있다[8,9,10]. 즉, 웹 서버 클러스터를 외부의 클라이언트들이 볼 때 마치 단일 서버로 보일 수 있도록 디스패처가 후위(back-end)의 서버들을 관리하는 기능을 구현하는 것이 중요하다는 것이다. 이때, 디스패처는 후위 서버들간에 부하(load)가 균형을 이룰 수 있도록 효과적인 세션분배정책(session distribution policy)을 수행할 수 있어야 한다.

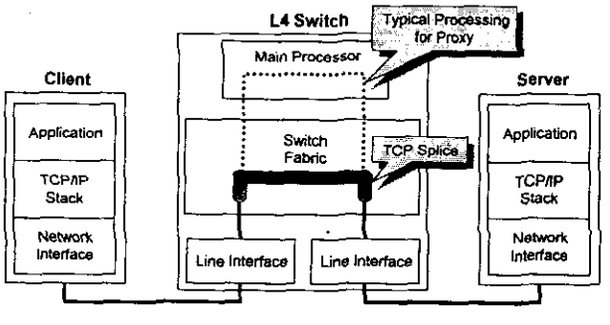
이러한 디스패처의 기능을 구현하기 위하여 현재 전세계의 많은 네트웍 장비공급 업체들이 솔루션을 제공하고 있는데, 상용화된 대표적인 제품들로 Cisco사의 Local Director[11], Zeus사의 Load Balancer[12], Nortel Networks사의 Alteon 780 Content Switch[13]등이 있다. 이들은 공통적으로 후위 서버들간에 부하균형을 유지하는 기술과 컨텍스트-인지형 디스패칭(context-aware dispatching) 기술을 구현하고 있다.

현재 네트웍 코어(core)가 제공하는 전송속도가 수십 Gb/s 급 이상이 되면서[4-7] 서버와 클라이언트 간의 통신에 대한 고속화 요구가 제시되고 있는 실정이다. 따라서, 시스템 설계자가 웹 서버 클러스터의 구조에서 부하균형만을 고려하는 것이 아니라 라인과의 인터페이스에 대한 고속화 구현도 함께 고려하여야 하는 상황이 되었다.

이러한 네트웍 고속화의 경향에 부합하도록 웹 서버 클러스터 환경에서도 기존의 후위 다중 서버들을 관리하는 기술을 바탕으로 하여 고속의 TCP 커넥션을 설정하고 관리하는 기술의 중요성이 부각되고 있다. 이를 위하여 다음 장에서 논의할 TCP Splicing과 같은 TCP커넥션 고속화 기법이 등장하게 되었다. 이를 활용하여 웹 서버 클러스터 환경에서 고속이며 특정목적을 갖는 세션들을 효율적으로 생성하고 관리하는 기술은 현재 전자상거래 서비스를 강화하는 핵심기술이 된다. 이러한 연유로, 본 논문에서는 전송성능 최적화를 고려할 대상으로 웹 서버 클러스터를 선택하게 되었으며 논의의 초점은 이러한 환경에서 클라이언트와 서버간의 데이터 전송 고속화를 구현하는데 맞추기로 한다.

3. TCP Splicing기술

앞 장에서 웹 서버 클러스터 환경에서 전송 고속화가 이루어짐에 따라 고속의 TCP 커넥션을 생성하고 유지하는 기술이 필요하게 되었다는 사실을 언급하였다. 본 장에서는IBM에서 제안한 TCP Splicing 기술에 대하여 소개하고 이 기술을 활용하여 고속의 TCP 커넥션을 설정하는 기법에 대하여 논의하고자 한다.



(그림 2) TCP Splicing 기능을 탑재한 디스패칭 시스템의 블록 개요도

3.1 동작원리

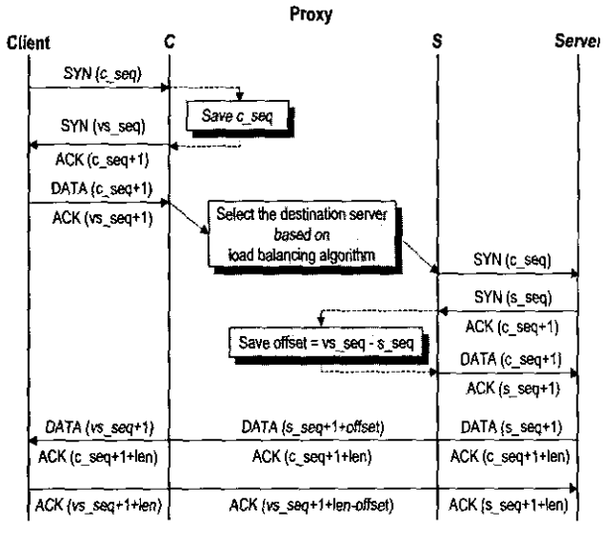
TCP Splicing이란 클라이언트나 서버와 같은 독립 시스템의 TCP/IP 스택에 적용되는 기술이 아니라, (그림 2)에서 보는 것처럼 클라이언트와 서버를 연결하는 프록시 서버((그림 2)에서 L4 switch에 해당)에서 TCP 커넥션을 구성하는 기술이다. 기존의 프록시 서버의 경우는 클라이언트와 서버간에 커넥션을 생성하기 위하여 클라이언트로부터 받은 패킷에 대한 헤더처리를 수행하고 패킷을 프록시의 주프로세서(main processor)에 상주하는 응용계층 프로세스까지 올리는 작업을 수행하는 방식을 취하였다. 이 방식을 이용하면 커넥션을 구성하기 위하여 주프로세서 내의 응용계층 소프트웨어를 다루기 때문에 다양한 응용서비스에 적합한 유연성이 있는 프록시 서버를 제공할 수 있는 장점이 있지만 패킷이 프록시를 통과하면서 여러 계층에서 빈번한 데이터의 복사가 일어나 궁극적으로 전송속도가 저하되는 취약점을 갖는다. 이러한 단점때문에 기존의 응용계층 프록시 서버는 백본망의 고속화를 통하여 전체적으로 네트웍 장비 가 고속화되고 있는 경향에 역행하는 솔루션이라고 볼 수 있다. 앞 장에서 설명한 것처럼 중소규모의 서버들을 클러스터링하는 서비스 환경에서 서버 클러스터군(郡)과 클라이언트 사이에 연결을 중재하는 프록시는 필수적인 네트웍 장비로 부각되고 있다. 이러한 프록시의 전송성능을 개선하는 것은 네트웍 서비스의 품질을 향상시키는 것과 일맥상통하게 된다. 이러한 전송성능 개선을 위하여 TCP Splicing이라는 기술이 제안되었는데[13,14,19,20], 기본적인 아이디어는 (그림 2)에서 보는 것처럼 클러스터 서버의 프록시로 사용되는 4계층 스위치(L4 Switch)의 라인 인터페이스 카드에서 클라이언트와 서버간의 TCP 커넥션이 직접 연결되도록 하여 전체 패킷 데이터 전달경로에서 주 프로세서를 거치는 과정이 제외되도록 하는 것이다. 이렇게 하면, 패킷 데이터가 프록시 스위치의 주프로세서로 전달되기 위하여 발생하는 데이터 복사과정이 제거됨으로써 클라이언트와 서버의 양 종단 간의 데이터 전송속도가 라인 속도를 유지할 수 있도록 시스템을 구성할 수 있게 된다.

3.2 TCP Splicing의 구현

앞 절에서 TCP Splicing의 역할 및 기본 개념에 대하여 알아 보았다. 이 절에는 앞에서 설명한 TCP Splicing의 기본 아이디어를 실제적으로 구현하는 방법에 대하여 알아보기로 한다.

TCP Splicing은 전체적으로 3단계의 과정을 거치게 된다(그림 3 참조). 1단계로 클라이언트와 프록시 간에 TCP 커넥션을 생성한다. 2단계에서는 서버와 프록시 간에 TCP 커넥션을 생성하는데, 이때 대상 서버의 선정은 기본적으로 부하균형(load balancing)방식[21]을 따른다고 가정한다. 항상 부하균형방식을 적용하는 것은 아니며, 클라이언트의 요청내용을 고려하여 연결할 대상 서버를 고려하기도 하는데 이러한 방식은 다음 절에서 고려하기로 한다. 마지막 3단계에서는 헤더변환(header translation)을

통하여 앞의 두 단계에서 생성한 두 개의 커넥션들 사이에 직접 연결이 이루어지도록 한다. 이렇게 되면, 생성된 커넥션이 종료될 때까지 클라이언트가 보낸 데이터는 프록시의 라인 인터페이스 카드에서 단순하게 헤더변환만을 통하여 서버로 보내지게 된다.



(그림 3) TCP Splicing에서 고속 커넥션을 위한 sequence number 변환 타이밍 다이어그램

따라서, TCP Splicing에서 효과적인 헤더변환이 핵심기술이라고 할 수 있는데, 이에 대한 기존에 제안된 방식[14, 15, 18]을 살펴보기로 한다. (그림 3)에 TCP Splicing을 위한 절차를 나타내는 타이밍 다이어그램을 나타내었다. 그림에서 보는 것처럼 클라이언트가 커넥션을 개시하기 위하여 SYN 패킷을 프록시로 보내게 된다. 이때, 클라이언트가 보내는 패킷의 TCP 헤더의 Flag 필드의 SYN 비트가 ON되게 되고, sequence number 필드에 임의의 32 비트의 수(c_seq)가 생성되어 설정되어야 한다. 이 sequence number는 현재의 커넥션에서 클라이언트가 보내는 패킷의 순서제어에 활용되게 된다. 이때, 바로 서버와의 커넥션이 생성되는 것이 아니기 때문에 프록시가 서버를 대행하여 클라이언트와의 커넥션을 수락하는 작업을 수행한다. 프록시가 서버를 대신하여 보내는 SYN 패킷을 위하여, 클라이언트가 수행한 것과 동일하게 sequence number(vs_seq)를 생성하게 된다. 이 sequnce number는 클라이언트의 IP주소(addr_c)와 포트번호(port_c)를 이용하여 아래의 식(1)과 같은 방식으로 생성할 수 있다.

$$vs\_seq = H(addr\_c, port\_c) \quad (1)$$

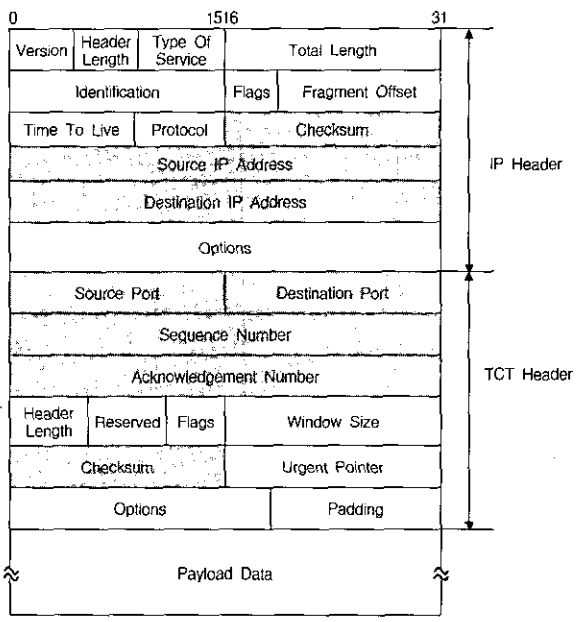
여기서 H는 32비트의 수를 생성하는 해쉬함수(hash fuction)를 나타낸다. 이 해쉬함수는 모든 라인 인터페이스 카드에 동일하게 구현되어 있어 주프로세서의 중재가 없어도 특정 클라이언트에 대하여 IP주소와 포트번호만 알면 어느 인터페이스에서도 동일한 sequence number를 생성할 수 있게 된다.

프록시가 대상서버를 대신하여 보낸 SYN 패킷에 대하여 클라이언트가 수락하는 ACK 패킷을 보냄으로써 클라이언트와 프록시 간의 커넥션 설정이 마무리되게 된다. 다음단계로 프록시와 서버사이의 커넥션을 설정하기 위하여 프록시는 대상서버를 부하균형 알고리즘에 의하여 선정하고 선택된 서버로 클라이언트를 대표하여 SYN 패킷을 보내게 된다. 이때, 프록시는 이전에 클라이언트로부터 SYN 패킷을 받았을 때에 저장하여 둔 sequence number(c_seq)값을 메모리로부터 읽어 서버로 보내는 SYN 패킷의 sequence number로 재설정하는 기능을 수행해야 한다. 클라이언트와 프록시 간에 수행되었던 three-way handshake 과정이 반복되는데, 차이점은 이때, sequence number 변환을 위한 offset값이 결정된다는 것이다. 즉, (vs_seq - s_seq)값에 해당하는 offset이 설정되어 서버와 연결되는 프록시의 라인 인터페이스 카드의 메모리에 저장된다. 따라서, (그림 3)에 나타난 것처럼 서버 측의 라인 인터페이스 카드에서 이 offset값을 기준으로 sequence number의 변환을 자동으로 해줌으로써, 이후의 패킷 전달이 프록시의 주프로세서까지 복사되지 않고 라인 인터페이스 카드에서 처리될 수 있게 되는 것이다.

앞에서는 sequence number의 변환에 중점을 두고 헤더변환의 동작흐름을 설명하였는데, 실제적으로 sequence number이외에도 변환되어야 할 필드로 IP헤더의 IP주소와 TCP 헤더의 포트번호 및 체크섬(check sum) 필드가 추가적으로 존재한다. (그림 4)에서 헤더변환과정에 변경되어야 할 대상을 정리하여 나타내었는데, 음영이 있는 필드는 모두 변환 대상이 된다. 클라이언트가 커넥션을 개시할 때 실제 연결될 서버의IP주소와 포트번호를 사용할 수 없으므로 프록시의 IP주소와 포트번호를 사용하게 된다. 그러나, 프록시의 중재 하에 대상서버가 선정된 후에는 실제 서버의 IP주소와 포트번호를 사용하여야 하므로 서버와 연결된 라인 인터페이스 카드에서 자동적으로 주소변환이 이루어져야 한다. 이런 과정에서 TCP헤더의 일부 필드 값이 변경되므로 체크섬 값도 이에 맞추어 자동적인 변경이 필요하게 된다.

이러한 동작원리에 기반한 TCP Splicing기술을 적용하여 수행한 성능시험의 결과가 여러 연구단체에 의하여 보고된 바 있다[14-18]. 이러한 결과들을 통하여 TCP Splicing에 기반한 TCP 커넥션 기술이 전송 웹서버 클러스터 환경에서 고속의 데이터 전송을 가능하게 함을 알 수 있으나, 이러한 결과는 서버에 요청되는 데이터의 컨텐트는 고려하지 않고 단순히 데이터 전송만을 고려한 결과이었다.

오히려, 획일된 부하균형 알고리즘이 연속된 HTTP 세션 처리나 보안 세션처리에서 서버의 캐쉬 히트율(cache hit rate)을 저하시켜 전체적으로 전송성능을 저하시키는 요인이 될 수 있다. 따라서, TCP Splicing기술에서 서버선정 알고리즘을 개선할 필요가 대두되게 되었다. 다음 장에서는 클라이언트가 요청하는 데이터의 컨텐트를 고려한 서버선정 알고리즘 개선 연구에 대하여 기술하기로 한다.



(그림 4) 헤더변환에 사용되는 IP헤더와 TCP 헤더의 필드

4. Content-Aware Dispatching 기술

4.1 컨텐트-인지 기술의 도입

앞장에서는 서버선정에 있어서 무조건적으로 부하균형만을 고려한 디스패칭 기술을 가정하였다. 부하균형을 극대화하기 위한 가장 이상적인 형태의 서버선정 알고리즘으로 라운드 로빈(round robin) 방식[22]을 들 수 있다. 라운드 로빈 방식을 이용한다면 연속적으로 발생하는 클라이언트의 요청을 매번 다른 서버가 서비스하게 될 것이다. 따라서, 라운드 로빈 방식을 서버선정 스케쥴링 방식으로 적용하는 경우는 모든 클라이언트의 요청이 서로 독립적이어서 캐쉬를 활용할 필요가 없다면 최적의 성능을 나타내겠지만, 이러한 가정은 실제의 네트웍에서는 의미가 없다. 대부분의 클러스터링된 서버들은 최근에 수행된 세션을 캐쉬에 보관하고 클라이언트의 요청이 있을 때 자신의 캐쉬정보를 이용하여 빠른 응답을 수행함으로써 전송성능을 향상시키고 있다. 이러한 웹 서버 클러스터 환경에서 라운드 로빈 방식으로 클라이언트에 대한 서비스를 수행할 서버를 선정하는 방법은 캐쉬의 히트율(hit rate)을 저하시켜 전체적으로 네트웍의 전송성능에 악영향을 미칠 수 있다. 그러므로, 디스패처가 클라이언트와 서버간의 커넥션을 형성할 때 서비스를 수행할 서버를 선정하는 알고리즘에 세션정보가 포함될 수 있도록 해야 한다. 즉, 현재 생성하려는 세션이 최근에 이미 서비스된 세션인지를 판단하여 캐쉬에 임시로 저장된 세션을 그대로 로드(load)하는 방법을 사용하기 위해서는 디스패처가 특정 서버로 세션을 라우팅하는 과정에서 세션정보도 라우팅 정보로 활용하여야 한다. 따라서, 무조건적으로 부하균형을 유지하지않고 세션의 재사용성(session reusability)을 고려하여 부하균형을 유지하는 방식으로 매 커넥션을 생성하는 것이 전송성능을 최적화하는데 필수적이게 된다.

이러한 컨텐트를 인지하고 이에 맞추어 요청을 분배하는 방식을 도입함으로써 얻어지는 성능개선효과는 다음과 같이 정리할 수 있다.

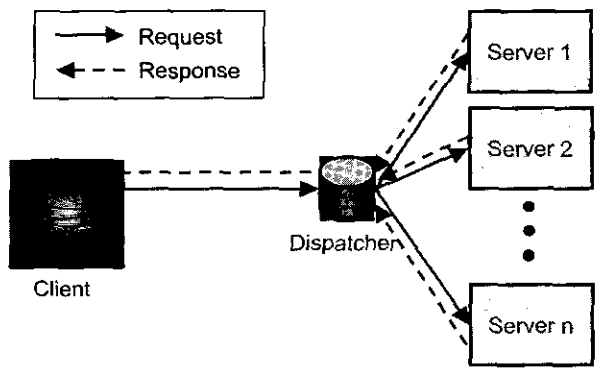
- 후위 서버의 캐쉬메모리의 히트율을 높임으로써 서버의 응답속도를 향상시킬 수 있다.
- 기존에 단일 서버의 데이터베이스에서 제공하던 서비스를 분할하여 다중 서버에게 나누어 제공하게 함으로써 확장성을 증대시킬 수 있다.
- 후위의 각 서버들을 특정 서비스에 맞게 특성화할 수 있다. 즉, 비디오와 오디오 서비스를 제공한다고 할 때 각 서비스별로 서버를 구분하여 사용함으로써 비디오 서버는 비디오 데이터 전송에 최적화될 수 있도록 구성할 수 있게 된다.

4.2 구현방식

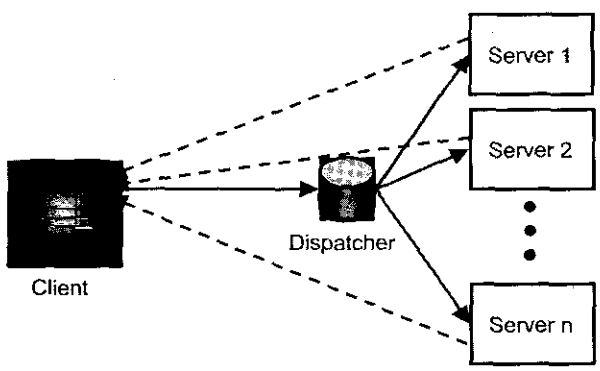
클라이언트의 요청을 처리하는 방식은 크게 (그림 5)에 나타낸 것처럼 relaying front end[16, 17], TCP handoff[15]의 두 가지 방식으로 구현된다.

Relaying front end 방식은 4장에서 설명한 TCP Splicing 방식에 예를 들어 LARD(Locality-Aware Request Distribution)[15]와 같은 컨텍스트-인지형 디스패칭 알고리즘을 적용하는 형태로 구현되게 된다. 여기서 예로 든 LARD는 Rice University에서 제안한 알고리즘으로 클라이언트로부터 요청이 발생하면 디스패처가 URL과 같은 컨텐트 정보를 기반으로 해당하는 후위 서버를 찾아 분배하고 실시간으로 서버의 부하를 측정하여 과부하가 발생하는 경우는 다른 서버로 부하를 분산시키는 것을 기본 아이디어로 하고 있다. 이러한 relaying front end방식은 디스패처가 부하분배에 대한 모든 책임을 지기 때문에 시스템 구성의 복잡도가 낮아지고 후위 서버들의 운영체제 커널에 구현된 기능을 변경할 필요가 없는 장점이 있지만 반대로 후위 서버로부터 클라이언트로 응답되는 모든 데

이터가 디스패처를 통과함으로 인하여 발생하는 오버헤드는 시스템의 성능을 저하시키는 요인으로 작용할 수 있는 단점이 있다.



(a) relaying front end



(b) TCP handoff

(그림 5) 컨텐트-인지형 디스패칭 기술

TCP handoff방식은 (그림 5b)에서 보는 것처럼 디스패처가 부하분배를 관리하지만 선정된 후위의 서버로부터 클라이언트로 보내지는 응답은 디스패처를 통과하지 않고 직접적으로 클라이언트에 연결되는 구조를 갖는다. 즉, 클라이언트로 보내는 응답 데이터의 전송에 대한 책임이 디스패처에서 서비스를 수행하는 후위 서버에게로 이동함으로써 relaying front end방식에서 문제시되었던 오버헤드를 줄일 수 있고 확장성이 강화된다는 장점이 있지만 후위의 각 서버가 이를 위하여 운영체제의 커널을 변경하여야 하고 시스템 구성이 단순하게 운영되지 않는 단점을 가지고 있다. 이 구조의 경우는 실제적으로 디스패처가 부하분배를 위한 작업을 수행하기 때문에 이론상 제시한 것만큼 확장성이 개선되지는 않는다는 실험결과가 보고된 바 있다[16, 17]. 따라서, 다음 장에서 제시할 보안 세션 성능최적화 방안에서는 기존의 relaying front end 방식에 기반을 두고 있다. 이렇게 함으로써 시스템 구성을 간단히 할 수 있고 새로운 알고리즘을 추가하여 성능을 개선할 수 있게 된다.

5. 웹 서버 클러스터 환경에서의 고속보안 세션서비스 제공방안

3장과 4장에서 고속의 TCP 커넥션을 생성하기 위하여 개발된 기술들을 알아 보았다. 이러한 컨텐트-인지형의 고속의 TCP 커넥션의 중요한 응용분야로 보안세션 기반의 VPN(Virtual Private Network) 서비스를 들 수 있다. 서론에서도 언급한 것처럼 전자상거래를 위한 웹 서버 시스템에서 발생하는 트랜잭션에 대하여 확실한 보안을 제공하는 것은 전자상거래를 가속화시키는 중요한 기술적인 요소가 되고 있다. 그러나, 이러한 보안기능을 현재 네트웍에 적용하고자 할 때 이를 어렵게 만드는 기술적인 요소로 암호화 및 인증 과정을 거침으로써 발생하는 전송지연(transmission delay)을 들 수 있다. 따라서, 보안세션을 사용하면서도 전송속도에 영향을 주지 않는 네트웍으로의 진화하기 위한 솔루션을 찾으려는 노력이 지속되고 있다.

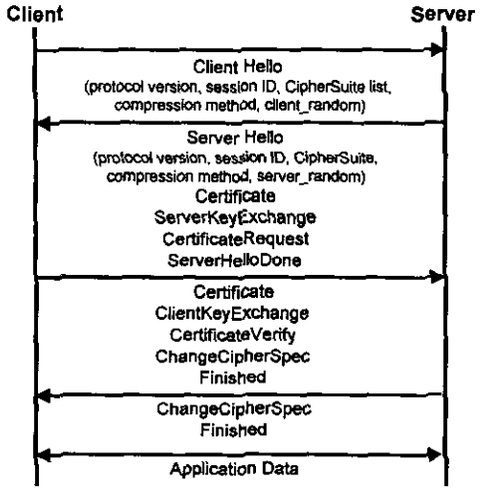
현재 보안세션을 제공하는 서비스로 산업계의 표준으로서 Netscape사가 제안한 SSL(Secure Socket Layer)[23]이 가장 널리 사용되고 있다. SSL은 TCP커넥션 위에서 동작하는 보안 서비스로서 HTTP, FTP(File Transfer Protocol)나 SMTP(Simple Mail Transport Protocol)와 같은 응용계층 서비스와도 원활하게 연동하면서 다양한 인증 및 암호화 기법을 제공하는 장점에 힘입어 현재 가장 대표적인 보안 소프트웨어로 인식되게 되었다. 이러한 영향력을 고려하여 IETF(Internet Engineering Task Force)는 SSL을 보안세션 서비스의 표준으로 채택하고 TLS(Transport Layer Security)[24]로 개명하여 표준화작업을 진행하고 있다.

그러나, 현재 표준화되고 있는 TLS 프로토콜은 웹 서버 클러스터 환경에서 고속의 전송을 보장하지는 못하고 있다. 본 장에서는 TLS 프로토콜의 메커니즘에 대하여 간단히 살펴보고, 웹 서버 클러스터 환경에서 TLS 프로토콜을 이용하여 보안서비스를 구성할 때 전송속도 저하를 막을 수 있는 개선안을 제안하고자 한다[26].

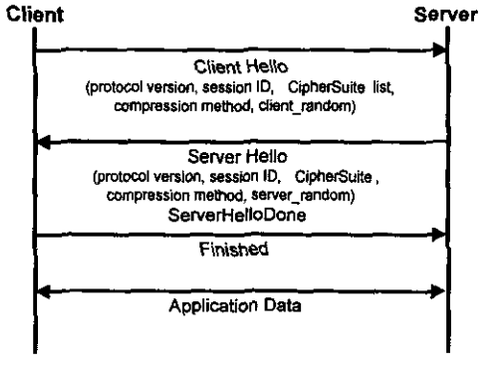
5.1 보안세션 재사용 효과

TLS 프로토콜은 TCP커넥션 위에 전송데이터를 일정 크기로 분할(fragmentation)하고, 암호화 및 인증과정을 수행하여 보안세션을 생성하는 레코드(record) 프로토콜을 하부 프로토콜로 갖고 있다. TLS 프로토콜을 사용할 때 모든 전송 데이터는 이 레코드 프로토콜을 통하여 TCP 커넥션을 이용하게 된다. 레코드 프로토콜이 보안세션을 생성하기 위하여 암호화 및 인증작업을 위한 보안키가 쌍방 통신객체 간에 미리 설정되어 있어야 한다. 이를 위하여 핸드쉐이크(handshake) 프로토콜이 별도로 존재하며,

(그림 6a)에서 보는 것처럼 완전한 핸드쉐이크 과정은 복잡한 과정을 거치게 된다. 특히, 클라이언트와 서버가 hello 메시지를 서로 교환한 후, 클라이언트로부터 서버로 보낸 메시지에 포함된 pre-master secret key를 서버가 해독하여 master secret key를 생성하는 과정은 많은 계산량을 요구하게 되어 시스템의 전송성능을 저하시키는 요소로서 작용하게 된다. 따라서, 매 커넥션마다 master secret key를 생성하는 것보다 커넥션이 해제되어도 특정 세션에 대한 master secret key를 세션 식별자별로 테이블화하여 저장하고 이를 재사용할 수 있다면, 시스템에 불필요한 부하를 줄일 수 있을 것이다. 즉, (그림 6b)에서 보는 것처럼 세션 재사용성을 높임으로써 전체 핸드쉐이크 과정이 간단해질 수 있다.



(a) 완전한 핸드쉐이크 과정



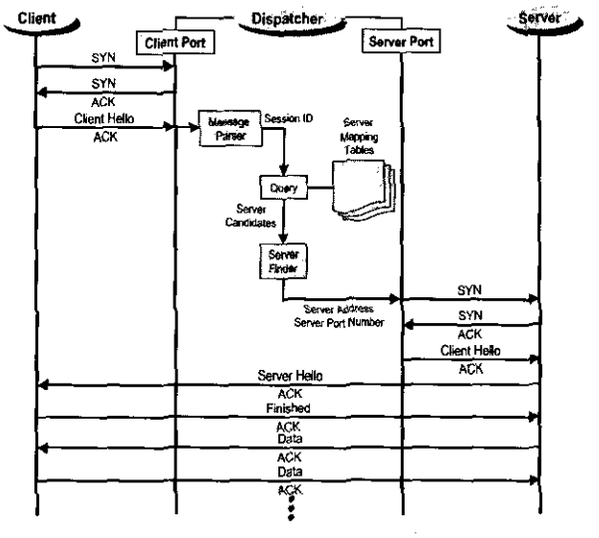
(b) 간이 핸드쉐이크 과정

(그림 6) TLS 핸드쉐이크 프로토콜

그러나, 세션 재사용성을 높여서 전송성능의 향상을 도모하기 위한 방법이 모든 네트워 환경에 적용되는 것은 아니다. 본 논문에서 고려하는 클라이언트-서버 네트워 구조인 웹 서버 클러스터 환경에서는 지나친 세션 재사용은 부하균형을 해치고 특정 서버에 심하게 부하를 걸어주게 되어, 네트워의 유휴자원(idle resource)이 있음에도 불구하고 네트워이 폭주하게 되는 극단적인 상황이 야기될 수도 있다. 따라서, 다음 절에서는 부하균형과 세션 재사용 간에 균형을 이룰 수 있는 방안을 고려하여 네트워의 클라이언트와 서버 양 종단간의 전송속도 저하를 최소화할 수 있는 핸드쉐이크 알고리즘을 제시하고자 한다.

5.2 TLS 핸드쉐이크 프로토콜 개선안

기본적인 아이디어는 앞에서 기술한 TCP Splicing기술과 컨텍스트 인지형 디스패칭 기술을 조합하여 특정시간 구간 내에는 세션을 재사용하지만 주기적으로 저장되어 있는 세션정보를 초기화해 줌으로써 장시간을 통하여 볼 때 각 서버에 걸리는 부하는 항상 균형을 이루고 있도록 하는 것이다. 이에 더불어, 주기적으로 세션정보를 초기화함으로써 PFS (Perfect Forward Securacy)[25]를 유지할 수 있게 된다.



(그림 7) 세션 재사용을 고려하여 전송성능을 개선한 TLS 핸드쉐이크 프로토콜 타이밍 다이어그램

제안하는 핸드쉐이크 프로토콜 개선안의 전체적인 플로우를 (그림 7)에 나타내었다. 그림에서 보는 것처럼TCP 커넥션을 설정하고 그 위에서 이루어지는 전체적인 데이터 교환은 TCP Splicing기법을 따른다. 보안세션 설정에 필요한 메시지이외의 데이터는 TCP Splicing으로 형성된 고속의 포워딩(forwarding) 경로를 통과하게 된다. 이렇게 함으로써 고속의 스위칭 시스템을 클러스터링된 서버들 간에 효과적인 디스패칭 서비스 제공자로 활용할 수 있다. 본 논문에서 제안하는 프로토콜 개선안은 클라이언트와의 데이터 교환에 적용될 서버를 선정하고 커넥션을 설정하는 과정의 개선에 있다. 즉, (그림 7)에서 보는 message parser

에서 server finder로 이어지는 과정의 최적화가 제안하는 아이디어의 핵심이 된다.

동작의 전체적인 흐름은 다음과 같다. 디스패처가 클라이언트로부터 *hello* 메시지를 받으면 *message parser*를 통과시켜 클라이언트가 요청하는 세션 식별자(session ID)를 추출해 낸다. 이때, 세션 식별자가 null이면 완전한 핸드쉐이크 과정을 거치고, 특정의 값을 가지면 그 값을 인덱스로 하여 서버 매핑 테이블(server mapping table)을 검색한다. 본 논문에서 제안한 알고리즘에서 검색결과는 유일의 대상서버를 제시하지 않는다. 이는 유일의 대상서버를 제시하게 하면 앞에서 설명한 것처럼 클러스터링된 서버들 간에 부하균형이 깨질 위험성이 존재하게 되기 때문이다. 따라서, 본 논문에서 제안한 알고리즘에서 리턴되는 결과는 대상서버 후보들을 다중으로 제공하는 방식을 취하고, server finder가 policy를 적용하여 이중에서 적합한 하나의 서버를 선정하는 방식을 취하였다. 이때, 적용하는 policy에 부하균형이 고려될 수 있도록 하였는데, 자세한 내용은 다음 절에서 설명하기로 하겠다. 이렇게 대상 서버를 선정하고 난 후, 디스패처의 서버포트 인터페이스는 헤더 변환을 통하여 클라이언트와 선정된 서버 간의 직접적인 커넥션을 만들어 주어 포워딩을 통한 고속의 데이터 전송을 구현한다.

5.3 부하균형을 고려한 서버선정 알고리즘

부하균형을 고려하기 위하여 서버 매핑 테이블을 (그림 8)과 같이 구성하기로 한다. 그림에 나타낸 테이블의 각 항목에 대하여 정의를 하면 다음과 같다. 첫번째 항목은 session ID라는 항목 타이틀을 가지고 있는데, 클라이언트와 특정 서버 간에 세션이 생성될 때 부여된 세션 식별자를 저장한 것으로써 서버 매핑 테이블을 검색하는 인덱스가 된다. sever IP address와 server port number는 그 세션 식별자를 가지고 클라이언트와 세션을 설정한 서버 응용 프로세스의 실제적인 IP주소와 TCP 포트번호를 나타낸다. timestamp 항목은 해당 세션이 종료된 시점의 디스패처 시스템의 시스템 시간을 저장한 것이다. 특히, 이 경우에 디스패처의 시스템 시간을 모든 세션에 적용하기 때문에 각 서버 간에 동기화를 위한 별도의 장비가 필요없게 되는 장점이 있다. 마지막 server state 항목은 현재의 해당 서버 프로세스의 상태를 나타내는데, BUSY는 현재 다른 클라이언트 프로세스에 의하여 점유되어 있음을 표시하고 IDLE은 유휴상태에 있음을 표시하게 된다.

이렇게 정의된 서버 매핑 테이블을 이용하고 관리하는 방법은 다음의 기본 알고리즘을 따른다. 세션 식별자로 테이블을 검색하여 얻어진 다중의 결과로부터 먼저 서버의 상태를 확인한다. 즉, BUSY상태를 가진 결과는 모두 후보에서 제외되게 된다. 그 다음에 대상 서버 후보들 중에 timestamp가 가장 작은 값을 갖는 - 즉, 가장 오래 전에 사용되었던 - 결과를 하나 선택하는 과정을 거친다. 이렇게 함으로써, 서버 프로세스들 간에 부하균형을 유지할 수 있게 된다. 더욱이, 기준시간 이전에 생성된 세션정보는 삭제하는 과정을 통하여 테이블의 내용을 주기적으로 갱신함으로써 테이블의 규모를 일정하게 유지하고 PFS도 부분적으로 제공할 수 있게 된다.

session ID	server IP address	server port number	timestamp	server state
⋮	⋮	⋮	⋮	⋮
122458	125. 212. 73. 5	80	20002	IDLE
135267	125. 212. 72. 9	80	20008	IDLE
122458	125. 212. 73. 4	80	20046	BUSY
261364	125. 212. 73. 2	80	20068	IDLE
⋮	⋮	⋮	⋮	⋮

(그림 8) 서버 매핑 테이블의 구성 예

디스패처가 서버 선정을 위한 스케줄링 및 클라이언트와 서버 간의 커넥션 생성의 책임을 지기 때문에 디스패처로 모든 트래픽이 집중되는 단점은 있지만 향후 고속 스위칭 기술로 구현된 라우터 장비들의 보급으로 이러한 단점은 쉽게 극복될 수 있으리라 판단된다. 따라서, 본 논문에서 제시한 고속의 보안 세션 제공방안은 구현자체가 간단하고 후위 서버와 클라이언트에 대하여 투명성(transparency)을 보장하기 때문에 앞으로 웹 서버 클러스터 기반의 전자상거래 시스템에 활용되기에 적합할 것으로 기대된다.

6. 결론 및 향후 연구 방향

본 논문에서는 웹 클러스터 서버 환경에서 고속의 TCP 커넥션을 생성하고, 그 위에 보안세션을 전송성능 측면에서 효율적으로 구성하는 방법에 대하여 논의하였다. 이러한 고속의 보안세션의 구성은 논문의 서두에서도 언급한 바 있듯이 인터넷 상에서의 전자상거래의 확산으로 인하여 해결해야 할 중요한 기술적인 문제로 대두되게 되었다. 본 논문에서는 세션 재사용성 증대와 부하균형의 유지라는 두 가지 구현 목표사이에 타협점(trade-off)을 찾는 방식으로 효율적인 보안세션 설계라는 문제 해결에 접근하였다. 클러스터링된 서버군(郡)의 전단(front end)에서 세션을 설정하고 서버에 부하분배정책(load distribution policy)을 수행하는 디스패처에 후위 서버들의 세션정보를 관리하는 테이블을

유지하게 함으로써 간단한 방식으로 부하균형을 유지하면서 세션 재사용성을 극대화하는 알고리즘을 구현할 수 있었다. 이는 앞으로 지속적으로 증가하는 전자상거래를 위한 응용 서비스를 고속의 네트웍 인프라에 효과적으로 접목시키는 중요한 인터페이스를 제공하는 기술이 되리라 기대되어진다.

본 논문에서는 디스패처 내에 존재하는 서버 매핑 테이블과 실제 후위 서버의 캐쉬에 존재하는 서버의 상태정보가 모든 순간에 일관성 있게(consistent) 유지되기 위한 방법에 대한 논의가 부족하였다. 이에 대한 연구를 지속적으로 수행할 예정이며, 이에 추가하여 이동 사용자(mobile user)를 위한 W-TLS(Wireless-Transport Layer Security)[27] 프로토콜에도 적합하도록 개선하는 방향에 대한 연구도 필요하다고 본다.

참 고 문 헌

[1] W. R. Stevens, *TCP/IP Illustrated Volume 1.*, Addison-Wesley, New York, 1996.

[2] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in Hardware at Memory Access Speeds," *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*, pp.1241-1248, 1998.

[3] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "The Tiny Tera : A Packet Switch Core," *IEEE Micro*, pp.26-33, 1997.

[4] A. Tantawy, O. Koufopavlou, M. Zitterbart, and J. Abler, "On the Design of a Multigigabit IP Router," *Journal of High Speed Networks*, pp.209-232, 1994.

[5] P. Newman, "IP Switching and Gigabit Router," *IEEE Communications Magazine*, pp.64-69, 1997.

[6] C. Partridge, *Gigabit Networking*, Addison-Wesley, New York, 1994.

[7] C. Partridge et al, "A 50-Gb/s IP Router," *IEEE/ACM Transactions on Networking*, pp.237-248, 1998.

[8] T. Schroeder, S. Goddard, and B. Ramamurthy, "Scalable Web Server Clustering Technologies," *IEEE Network*, pp.38-45, 2000.

[9] E. Levy-Abegnoli, A. Iyengar, J. Song, and D. Dias, "Design and Performance of a Web Server Accelerator," *IEEE INFOCOM'99*, pp.135-143, 1999.

[10] Web Server Farm Performance, White Paper, Arrowpoint Communications, 1998.

[11] Cisco Local Director, Technical White Paper, Cisco systems, 1998.

[12] Improving Web Server Performance with Zeus Load Balancer, Technical White Paper, Zeus Technology, 2000.

[13] Alteon 780 High Density Content Switch Data Sheet, Nortel Networks, 2001.

[14] D. A. Maltz, and P. Bhagwat, TCP Splicing for Application Layer Proxy Performance, *Journal of High Speed Networks*, pp.225-240, 1999.

[15] D. A. Maltz, and P. Bhagwat, Improving HTTP Caching Proxy Performance with TCP Tap, *IBM Research Report RC21147*, 1999.

[16] V. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, Locality Aware Request Distribution in Cluster-based Network Servers, *Architectural Support for Programming Languages and Operating systems*, pp.1-12, 1998.

[17] M. Aron, D. Sanders, and P. Druschel, Scalable Content-aware Distribution in Cluster-based Network Servers, *Proceedings of the 2000 USENIX Technical Conference*, 2000.

[18] O. Spatscheck, J. S. Hansen, J. H. Hartman, and L. L. Peterson, Optimizing TCP Forwarder Performance, *Technical Report 98-01, Dept. of Computer Science, University of Arizona*, 1998.

[19] A. Cohen, S. Rangarajan, and H. Slye, On the Performance of TCP Splicing for URL-aware Redirection, *Proceedings of the 2nd USENIX symposium on Internet technologies and Systems*, 1999.

[20] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, Fast and Scalable Layer Four Switching, *SIGCOMM*, 1998.

[21] V. Kumar, A. Grama, and V. N. Rao, Scalable Load Balancing Techniques for Parallel Computers, *Journal of Distributed Computing*, pp. 60-79, 1994.

[22] S. Keshav, *An Engineering Approach to Computer Networking*, Addison-Wesley, New York, 1997.

[23] Netscape Communications, SSL 3.0 Specification.

[24] T. Dierks, and C. Allen, "The TLS Protocol Version 1.0," RFC 2246, 1999.

[25] N. Doraswamy, *IPSec : The new Security Standard for the Internet, Intranets, and Virtual Private Networks* , Prentice Hall, Upper Saddle River, 1999.

[26] S.-E. Jin, T.-I. Kim, and H.-H. Lee, Improved TLS Handshake Protocol For Scalable Web Server Cluster, *International Conference on Advanced Communication Technology*, pp.841-844, 2001.

[27] S. Blake-Wilson, and M. Nystrom, Wireless Extensions to TLS, draft-ietf-tls-wireless-00.txt, 2000.

진 승 의

e-mail : jinse@etri.re.kr

1998년 한양대학교 전자공학과 졸업 (공학사)

2000년 한국과학기술원 전자전산학과 (공학석사)

2000년~현재 한국전자통신연구원 연구원 재직

관심분야 : 인터넷 보안, 고속 라우터, 광 네트워크

김 태 일

e-mail : ttikim@etri.re.kr

1998년 숭실대학교 전자계산학과(공학사)

1992년 정보처리기술사 취득

1983년~현재 한국전자통신연구원 책임연구원, 라우터 관리팀장

관심분야 : 고속 라우터, 지능망, 망관리, 개방형 네트워크

이 형 호

e-mail : holee@etri.re.kr

1977년 서울대학교 공업교육과 전자전공 졸업(공학사)

1979년 한국과학기술원 전기및전자공학과 (공학석사)

1983년 한국과학기술원 전기및전자공학과 (공학박사)

1984년~1986년 미국 AT&T Bell 연구소 방문 연구원

1996년~1998년 충남대학교 공과대학 전자공학과 겸임교수

1995년~1998년 대한전자공학회 회지편집위원장

1996년~1998년 대한전자공학회 전자교환연구회 전문위원장

1996년~1999년 IEEE ComSoc APB MDC의장

1998년~현재 대한전자공학회 이사, 회지담당 상임이사

1998년~현재 통신위원회 전문분과위원회 전문위원

1999년~현재 한국통신학회 교환 및 라우팅 연구회 위원장

2000년~현재 IEEE 대전 Section 부의장

2000년~현재 한국 이더넷 포럼 의장

2000년~현재 한국정보과학회 평의원

1983년~현재 한국전자통신연구원 소속 TDX개발단 신호방식 개발실장, 지능망교환기개발실장, 교환기술연구단 소프트웨어종합실장, 계통연구부장, 교환전송기술연구소 교환시스템연구부장 등 역임

현 재 한국전자통신연구원 네트워크기술연구소 라우터기술연구부장, 책임연구원

관심분야 : 유무선 고속LAN, 고속 라우터, 광 인터넷, 개방형 통신망, 유무선 ATM