웹서버 네트워킹에서의 트래픽분산 처리 시스템 구현

박길철* · 성경** · 김석수*

Traffic Distributed Processing System Implementation on the Web Sever Networking

Gil-cheol Park* · Kyung Sung** · Seok-soo Kim*

본 연구는 과학기술부 지역협력연구사업 (R12-2003-004-00008-0)지원으로 수행되었음.

요 약

본 논문에서는 Communication networking 개선을 위한 트래픽분산 시스템구현을 위한 2개의 패키지(패킷 캡춰와 라운드로빈 테스트 패키지)를 구현하였으며, 이러한 구현 패키지 S/W에 의해 본 연구실험의 가상연결구조(데이터 생성기, 가상서버, 서버1,2,3) 에서 발생하는 패킷의 량을 측정하였다. 즉, 각 서버1,2,3에 트래픽 분산여부를 파악할 수 있었으며, 구현된 트래픽 분산 시스템의 기능으로는 데이터의 수신량, 패킷량 표현, 패킷량 그래프 표현, 라운드 로빈 테스트, 시스템 모니터링기능이 있다.

본 구현시스템을 통한 실험결과, 들어오는 데이터의 크기가 많이 차이나지 않는 이상 라운드 로빈 알고리즘은 확실한 형태의 트래픽 분산을 가능하게 해주었다. 그리고 일부분에서 오차가 심한경우도 있지만 횟수를 거듭하고 테스트가 장기화 될수록 오차는 현저히 줄어들었다.

ABSTRACT

This paper introduces implementation of a traffic distributed processing system on the Web Sever Networking. The study used two software packages (Packet Capture and Round-Robin Test Package) to check packet quantity from Virtual Network Structure (data generator, virtual server, Server 1, 2, 3), and could find out traffic distribution toward Server 1, 2, and 3. The functions of implemented Round-Robin Load Balancing Monitoring System include Round-Robin testing, system monitoring, and graphical indication of data transmission/packet quantity (figures & diagram).

As the result of the study shows, Round-Robin Algorithm ensured definite traffic distribution, unless incoming data loads differ much. Although error levels were high in some cases, they were eventually alleviated by repeated tests for a long period of time.

키워드

(웹서버, 트래픽, 분산처리, 라운드로빈, 부하균형)

*한남대학교 **목원대학교 컴퓨터교육과

접수일자 : 2004. 2. 20

Ⅰ. 서 론

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 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.

중소규모의 다중 서버들로 구성된 웹 서버 클러스터를 수용한 네트웍 구조는 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.

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 이수가 되고 있다. [4,5,6]. 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 수행 할 수 있어야 한다.

이러한 네트웍 고속의 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 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.

즉, TCP Splicing이란 클라이언트나 서버와 같은 독립 시스템의 TCP/IP 스택에 적용되는 기술이 아니라, 클라이언트와 서버를 연결하는 프록시 서버에서 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.

Ⅱ. 웹서버 클러스트의 부하균형

현재 웹 환경에서의 정보 보안을 요구하는 정보들이 무수히 많이 존재한다. 예를 들어 개인 정보 중에서도 주민등록번호, 카드번호, 등 보안을 필요로 하는 정보들 있는데 이들을 암호화 하기 위해서는 SSL이 가장 널리 사용되고 있다.[7] SSL은 TCP 커넥션 위에서 동작 하는 보안 서비스로서 HTTP(HyperText Transfer Protocol) , FTP(File Transfer Protocol), 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 보안 세션 서비스의 표준으로 채택하고 TLS(Transport Layer Security)로 개명하여 표준화 작업을 진행하고 있다. 그러나, 현재 표준화되고 있는 TLS 프로토콜은 웹 서버 클러스터 환경에서 고속의 전송을 보장하지 못하고 있다.

보안 세션을 생성하기 위하여 보안키가 쌍방 통신객체 간에 미리 설정 되어 있어야 한다. 이를 위하여 핸드쉐이크 프로토콜이 별도로 존재하며, 여기에 사용 되어지는 pre-master secret key를 서버가 해독하여 master secret key를 생성하는 과정은 많은 계산량을 요구하게 되어 시스템의 전송성능을 저하시키는 요소로서 작용하게 된다. 따라서 매 커넥션마다 생성하기 보다는 재사용을 하는 편이 전송속도 향상에 도움을 준다. 그러나 세션 재사용성을 높여서 전송성능의 향상을 도모하기 위한 방법이 모든 네트워크 환경에 적용되는 것은 아니다. 클러스터 환경에서는 지나친 세션 재사용은 부하균형을 해치고 특정 서버에 심하게 부하를 걸어주게 되어 네트워크가 폭주하는 극단적인 상황이 야기될 수 있다. 따라서 부하 균형과 보안 세션 재사용 간에 균형을 이룰 수 있는 방안을 고려하여 전

체 네트워크의 전송속도 저하를 최소화할 수 있는 핸드쉐이크 알고리즘이 있어야 한다.

클라이언트의 요청을 처리하는 방식은 크게 Relaying front end[6,7]와 TCP handoff의 두가지 방식으로 구현된다. 먼저 Relaying front end방식의 한 예를 보면 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 한다. TCP handoff방식은 디스패처가 부하분배를 관리하지만 선정된 후위의 서버로부터 클라이언트로 보내지는 응답은 디스패처를 통과하지 않고 직접적으로 클라이언트에 연결 되는 구조를 갖는다. 여기에서 제시하는 보안 정책은 위의 두가지 방법을 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. 먼저 이 보안정책을 적용하기 위하여 위의 클라이언트 요청 처리방식을 구현해야만 할 것이다.[8]

최근 네트워크에 연결된 사용자 수의 증가와 그에 따른 서비스들의 증가는 그간의 전자적 기술의 발달이나 통신장비 기술의 발달로는 사용자의 요구를 만족시킬 수 없게 되었다. 이에 대한 해결책으로 최근 데이터를 셀 단위로 다루는 Load Balancing[9,10]이 제안되고 있다. 하지만 셀 단위의 스케줄링은 스위칭 후 데이터를 다시 재조립해야 하는 추가의 작업을 더 필요로 하게 된다. 패킷을 고정길이가 아닌 가변길이 패킷으로 스위칭 하게되면 셀 단위 스케줄링 방법에서의 재조립 단계를 생략하게 되어 트래픽에 대한 처리율을 높 일 수 있다.

1) 구현 목표

- 서버의 트래픽을 분산하여 좀더 안정적인 서비스를 제공한다.
- 여러 컴퓨터에 작업을 균등하게 분산시킴으로써 한 대의 컴퓨터를 사용하는 것에 비해 짧은 시간에 사용자가 원하는 작업을 수행하고자 하는 것이다. 수십대의 컴퓨터를 사용한다고 하더라도 주로 한 대의 컴퓨터에서 작업이 이루어 진다면 원하는 성능 향상을 얻을 수 없을 것이기 때문에, 작업을 "균등"하게 분산시키는 것이 중요하다.
- 작업의 균등한 분배를 위해서 클라이언트와 웹서버 사이의 가상 서버를 만든다. 클라이언트가 요청하는 모든 작업은 가상서버를 통해서 작업하게 될 서버가 정해지게 된다. 이때 클라이언트는 자신이 어떤 서버가 정해지게될지 알지못하게 된다.
- 분석된 트래픽을 한눈에 볼 수 있게 윈도우를 통한 GUI 환경을 제공하는 소프트웨어를 구현한다.

2) 구현기능

- 네트워크 트래픽을 제어하기 위해 전송 프로토콜 및 어플리케이션 계층의 프로토콜을 할 수 있어야 한다.
- 패킷을 가변 길이로 가정 후 Round Robin 알고리즘을 이용하여 트래픽을 제어한다.
- 분석된 트래픽을 한눈에 파악할 수 있게 그래프로 표현해야 한다.
- 하나의 소프트웨어를 구현해서 윈도우를 통해 사용자가 알아보기 쉬운 GUI환경으로 출력한다.
- 앞에서의 이론을 바탕으로 가상 서버를 구현한다. 작업을 요청한 클라이언트는 가상 서버를 통해서 어떤 서버로 연결될 것인지 결정되어 진다. 클라이언트 자신이 연결될 서버를 알지 못하고 가상 서버에 의해 연결되어진 서버에서 요청한 작업을 수행한다.

III. 기능모듈

시스템에 장착된 데이터의 송·수신을 담당하는 NIC(Network Interface Card)가 자기 자신의 시스템으로 오는 패킷만이 아닌 동일 네트워크 상에서 브로드캐스트되는 모든 패킷을 받아들인다는 사실을 기초로 하여 패킷을 캡쳐하고 캡쳐된 패킷의 헤더 정보를 분석하여 패킷의 유량을 그래프로 출력하고 보다 자세한 정보를 출력한다. 패킷을 수집하는 패킷 캡쳐 모듈과 분석모듈 및 출력모듈로 구성되는데 첫째, 패킷 캡쳐 모듈은 동일 네트워크 상에서 패킷들을 실시간으로 캡쳐 하여 분석 모듈로 전달되는데 Libpcap(Portable Packet Capturing Library) 라이브러리를 사용한다. 둘째 출력모듈은 이렇게 분석된 패킷에 대한 정보에서 프로토콜에 따라 유량(크기)을 그래프로 출력하고 보다 자세한 정보를 보여준다.

이 프로그램은 패킷을 캡춰하는 라이브러리를 제공하는 프로그램으로써 패킷 모니터링 하는데

반드시 필요한 라이브러리이다. 이 라이브러리는 다음과 같이 구성되어져 있다.

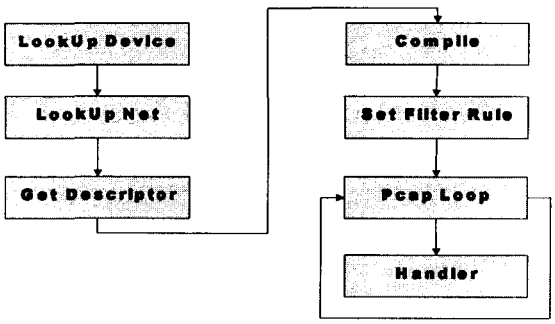

그림 1. Libcap 구조도
Fig. 1 Libcap Structure Diagram

(1) Lookup Device
실제 패킷 캡쳐에 사용되는 네트웍 인터페이스 카드(NIC)의 디바이스 명을 검색하기 위한 작업이다. 함수로는 pcap_lookupdev() 함수를 사용한다. 이 함수는 현재 시스템에서 네트워크와 직접 연결되어 있는 NIC 디바이스명의 문자열을 반환하는데 pcap_openlive() 함수와 pcap_lookupnet()함수의 전달 인자로 사용한다.

(2) Lookup Net
단계 (1)에서 검색된 NIC의 디바이스 명에 해당하는 네트웍 아이디(NetID)를 알아내기 위해 사용한다. 사용되는 함수는 pcap_lookup() 함수이며 전달 인자로는 NIC의 디바이스 문자열이다.

(3) Get Discriptor
패킷 캡쳐를 위한 디스크립터(Discriptor)를 얻는 과정이다. 이 과정에서는 pcap_ open_live()함수가 사용되는데 5개의 전달 인자가 필요하다. 첫 번째 전달인자는 NIC의 디바이스 문자열이고, 두 번째는 캡쳐 할 패킷의 크기를 지정하는데 SNAPLEN이라 한다. 이 크기를 지정하는 이유는 패캣의 최대 크기는 MTU를 고려해서 1514Bytes 이다. 그러나 패킷의 헤더부분은 처음 68바이트까지 이며 이 후부터는 데이터 부분이 되는데 패킷의 정보를 알아 내는데는 헤더부분만 있으면 되기 때문에 일반적으로 이 크기는 "68"을 지정한다. 세 번째는 PROMISCOUS 모드를 지정하는 부분인데 이 곳에는 "1"을 지정한다. 네 번째는 패킷 캡쳐의 타임아웃(Time Out)값을 지정하고, 마지막으로 에러가 발생하였을 경우 에러 내용을 저장할 버퍼를 지정한다.

(4) Compile 및 Set Filter Rule
이 단계는 룰셋을 지정하는 부분인데 Libpcap에서는 이 단계에서 지정된 룰의 규칙을 따른다.

(5) Pcap Loop 와 Handler
Libpcap의 마지막 단계는 캡쳐를 하는 단계와 캡쳐된 패킷을 처리하는 단계이다. 패킷을 캡쳐하기 위해서는 pcap_loop() 함수를 사용하는데 이 함수에도 역시 4개의 전달 인자가 필요하다. 첫 번째 전달인자는 (3)단계에서 얻어진 패킷 캡쳐 디스크립터를 적어주고, 두 번째는 캡쳐 할 패킷의 수를 지정하는데 "-1"의 값을 적어주면 무한루프를 돌면서 패킷을 캡쳐 한다. 세 번째는 핸들러(Handler) 함수의 포인터를 지정해 주는데 pcap_loop() 함수는 패킷을 캡쳐 할 때마다 지정된 핸들러(Handler) 함수를 호출해 패킷을 처리한다. 마지막 인수는 사용자 정의 인수로서 일반적으로 "NULL"을 명시한다.

(6) 서버에 존재하는 네트워크 장치와 패킷 캡쳐
서버에 존재하는 네트워크 연결장치 랜 카드 등을 검출하고 이 장치가 사용하는 ip주소와 서브넷 마스크 등을 알아내고 이 장치에 들어오는 패킷을 검출한다. 이 때 특정 장치와 특정 프로토콜을 설정할 수 있다.

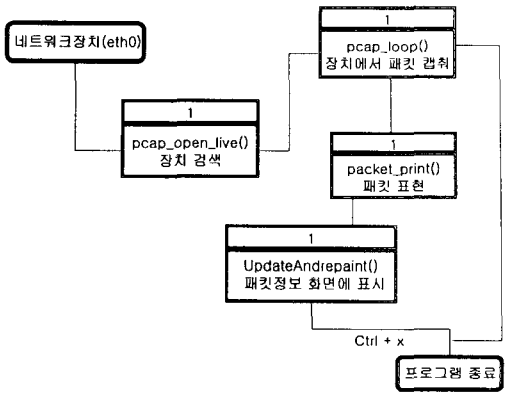

그림 2. 자료 흐름도
Fig. 2 Data Flow Diagram

IV. 라운드로빈 모니터링시스템

4.1 시스템구조 개요

(1) 데이터 생성기를 통해 보내지는 임의의 데이

터의 헤더파일을 분석하고, 그 결과를 생성기의 모니터를 통해 보여준다.

(2) 분석된 파일들을 가상서버에게 전달하고 Round Robin 알고리즘을 구현하여 웹 서버에 분배한다.

(3) 분배된 파일들을 다시 분석하여 그 결과를 각 서버의 모니터를 통해 보여준다.

4.2 시스템 구조도(리눅스 클러스터 구성)

리눅스 가상 서버[11]는 리눅스 운영체제를 기반으로 고속 네트워크에 연결된 서버들의 클러스터로 성능이 뛰어나고 가용성이 높은 서버를 쉽게 구축할 수 있다[12].

그림 3은 일반적인 가상 서버의 구조로서 하나의 부하 분산기와 여러 대의 리얼 서버를 가진다. 리얼 서버들은 동일한 서비스를 운영하도록 구성되며, 서비스 내용은 각 서버의 지역 디스크에 복제되거나, 분산 파일 시스템으로 공유함으로써 서비스를 제공한다. 부하 분산기는 리얼 서버의 서비스 내용에 대하여 클라이언트의 요청이 있을 때마다, 적절한 스케줄링 방법(예, Round Robin)에 의해서 상호 연결되어 있는 리얼 서버에 클라이언트의 요청을 발송한다.

4.3 시나리오 및 제약조건

(1) 시나리오

- 데이터 생성기를 통해 들어오는 패킷 정보를 분석할 가상 서버를 구축한다.
- 분석된 패킷 정보를 윈도우 형식으로 표현해 줄 소프트웨어를 개발한다.
- 최종 실험으로 데이터 생성기를 통해 입력되는 패킷을 전체 패킷량 TCP, IP, ARP, 그 밖의 프로토콜들을 가상서버에서 분석하여 다시 데이터 입력 컴퓨터에서 보여줄 수 있는지를 실험한다.
- 덧 붙여 가상서버가 분석한 데이터들을 각 서버 1,2,3에게 뿌려주었을 때 어떤 값이 돌아오는지도 알아본다.

(2) 제약조건

- 실 서버 환경에서의 실험을 위해 리눅스 운영체제가 필요하다.
- Round Robin 테스트를 위해 각각의 pc를 하나의 네트워크 환경으로 묶어줄 switching Hub가 필요하다.
- 패킷 데이터의 헤더파일을 분석해서 보여주기 위한 데이터 구현 필요하다.

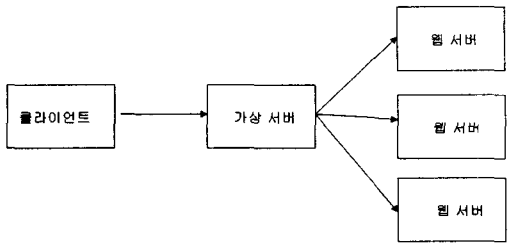

그림 3. 가상 연결 구조
Fig. 3 Virtual Connection Structure

V. 시스템구현 및 기능모듈

본 논문에서는 라운드 로빈 알고리즘을 적용한 부하균형을 하기위한 다음과 같이 2개의 패키지를 개발하고 각 패키지는 패킷 캡춰와 라운드로빈 테스트 패키지로 명하였다.

5.1 패킷 캡춰 패키지

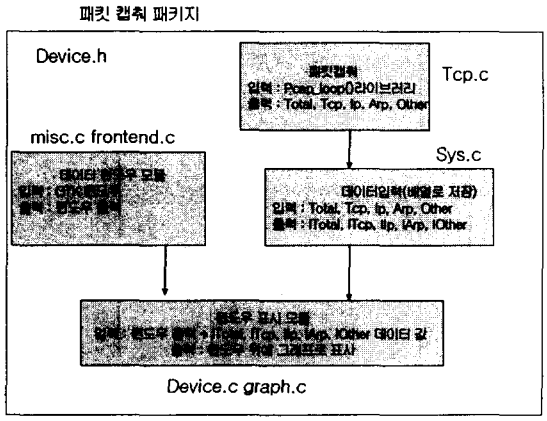

그림 4. 패킷 캡춰 패키지
Fig. 4 Packet Capture Package

패킷 캡춰 패키지는 총 2개의 모듈로 구성이 되며 각각 패킷 캡춰 모듈과 그래픽 모듈이라 명한다. 패킷 캡춰 모듈은 패킷의 량을 측정하는 부분으로 프로토콜 별로 텍스트로 데이터를 출력하고, 그래픽 모듈은 윈도우 창을 생성하고 패킷 캡춰 모듈로부터 넘겨 받은 데이터를 저장하고 그래프로 표시해 주는 역할을 한다.

5.2 라운드 로빈 테스트 패키지

라운드로빈 테스트 패키지는 총 3개의 모듈로

구성이 되며 각각 데이터 생성기 모듈, 가상 서버 모듈, 데이터 수신기 모듈이라 명한다. 데이터 생성기 모듈은 랜덤 값의 데이터를 생성하는 것으로 가상 서버 모듈에 전송하는 역할을 한다.

가상서버 모듈은 데이터 생성기로부터 받은 데이터를 라운드 로빈 알고리즘에 의해 분산시켜 각각 데이터 수신기에 전송하는 역할을 한다. 데이터 수신기 모듈은 가상서버 모듈에서 들어오는 데이터를 파일로 저장하여 비교 분석하는 역할을 한다.

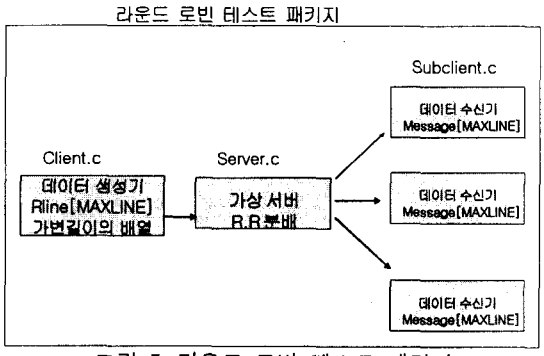

그림 5. 라운드 로빈 테스트 패키지
Fig. 5 Round Robin Test Package

5.3 인터페이스

1) 기본기능

- 라운드 로빈 테스트 : 라운드 로빈 알고리즘에 의해 네트워크의 트래픽이 분산되어지는 과정 표현
- 시스템 모니터링 : 패킷 모니터링을 하는 것을 말하며 시스템에 Total, TCP, IP, ARP, Other 이 5가지 종류의 프로토콜에서 들어오는 패킷의 량을 측정하는 프로그램으로 총 5가지의 버튼과 1가지의 메뉴가 있다.

2) 윈도우/메뉴/아이콘

- 윈도우 : 리눅스 환경에서 실행화면은 주로 텍스트이다 그래프를 표현하기 위하여 GTK를 이용한 리눅스 윈도우를 사용하였으며 여기에는 1개의 메뉴와 5개의 버튼이 들어가고 100픽셀로 적용하여 가로세로 100단계로 표현이 가능하다 실제 패킷의 량을 그 최대 크기를 가늠할 수 없지만 적은 경우 10단위부터 많은 경우에는 100만이 넘을 수도 있다 실제 동시 접속자 수가 100명 정도 되어지는 사이트에서 보통 30만 정도의 패킷량을 파악할 수 있었다. 이러한 기복 때문에 그래프 표현은 정확하게 표현할 수 없고 같은 상황에서의 비교 파악만이 가능하다. 본 S/W에서는 비교파악만 가능하면 되므로 이정도만 표현 하도록 했으며 추후 기능이 요구 되면 색으로 나누어 표현하는 방법을 사용하여 좀더 정확한 측정을 할 수 있도록 하겠다.
- 메뉴 : 종료 메뉴 윈도우 창을 종료시킴
- 아이콘 : 5가지 버튼을 두고 색으로 구별하여 5종류의 프로토콜 량을 측정함

3) 고급 기능 : 5가지 종류의 프로토콜(Total, TCP, IP, ARP, Other)의 량을 표현 할 수 있도록 하여 어떠한 프로토콜들이 많이 발생을 하는 지 파악을 할 수있다.

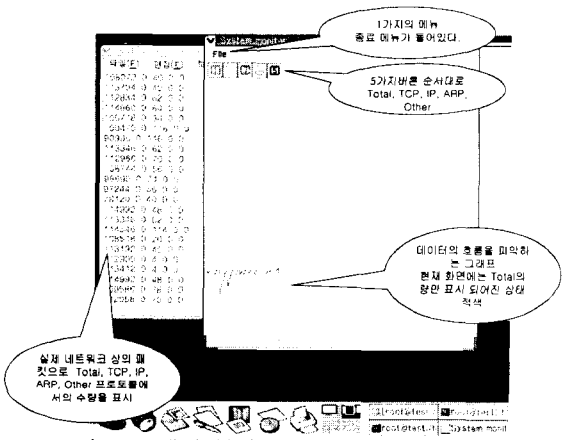

그림 6. 트래픽 분산처리 시스템의 실행화면
Fig. 6 Execution Window of Traffic Distrubuted Processing System

VI. 연구 결과 및 분석

다음 표들은 각각 데이터 생성기, 가상서버, 서버1,2,3 에서 발생하는 패킷의 량을 측정한 것으로 서버1,2,3에게 트래픽을 분산시키는지 여부를 파악할 수 있다

표 1. 각 클라이언트에 10초 30초동안 데이터전송
Table 1. Sending Data to Each Clients during 10~30 Second

<단위 : Byte>

클라이언트	10초 동안 데이터 전송		
client1	146526	146157	141433
client2	146776	127916	340939
client3	144901	141319	147129

클라이언트	30초 동안 데이터 전송		
client1	341379	240173	352137
client2	340939	243058	351996
client3	342612	359901	354563

표 2. 각 클라이언트에 1분동안 데이터 전송
Table 2. Sending Data to Each Clients during one minite

<단위 : Byte>

클라이언트	1분 동안 데이터 전송(a)				
client1	718342	732062	697351	702178	758586
client2	730943	708527	697549	703640	749016
client3	721741	703673	688999	701005	747809

클라이언트	1분 동안 데이터 전송(b)				
client1	693862	726317	696344	724229	675995
client2	673256	711579	716269	719180	683513
client3	676327	708681	705556	697063	691806

표 3. 표 1,2의 평균 데이터전송
Table 3. Data Sending Average of Table 1,2

<단위 : Byte>

클라이언트	10초	30초	1분
client1	144705	311229	712526
client2	205210	311997	709347
client3	144449	352358	704266

실험결과와 같이 들어오는 데이터의 크기가 많이 차이나지 않는 이상 라운드 로빈 알고리즘은 확실한 형태의 트래픽 분산을 가능하게 해준다. 일부분에서 오차가 심한경우도 있지만 횟수를 거듭하고 테스트가 장기화 될수록 오차는 줄어든다.

실험결과에서 각 서버1,2,3 에 걸린 패킷 량은 22000~25000 사이의 값으로 구성이 되어 있으며 실제 데이터 생성기에서 보낸 65000~70000 사이의 값을 확인해 볼 경우 상당히 정확히 분산되어지는 것을 볼 수 있다. 그러나 가상 서버에 걸린 패킷의 량을 보면 데이터 생성기에서 발생한 패킷 보다 훨씬 많은 90000~110000 사이의 값을 볼 수가 있다. 데이터 생성기에서 발생하는 패킷의 량의 거의 2배가 되는 것으로써 가상서버가 받아들이는 것과 보내는 패킷의 량이 될 것이라고 추측해 볼 수 있다. 또 한 여기에서 각 서버 1,2,3에 걸린 패킷의 량은 단지 들어온 값에 불과하므로 이를 처리하고 다시 가상서버에 보내는 것을 생각 했을 경우 가상서버에도 상당히 많은 량의 트래픽이 발생할 것이라 예상한다. 본론은 필요에 따라 3-4 개의 장으로 편집할 수 있습니다.

VII. 결 론

최근 네트워크에 연결된 사용자 수의 증가와 그에 따른 서비스들의 증가는 그간의 전자적 기술의 발달이나 통신장비 기술의 발달로는 사용자의 요구를 만족시킬 수 없게 되었다.

본 논문에서는 Communication networking 개선을 위한 트래픽분산 시스템구현을 위해 라운드 로빈 알고리즘을 적용한 부하균형기법을 보이고 있다. 이를 위하여 2개의 패키지(패킷 캡춰와 라운드로빈 테스트 패키지)를 구현하였으며, 패킷 캡춰 모듈은 패킷의 량을 측정하는 부분으로 프로토콜별로 텍스트로 데이터를 출력하고, 그래픽 모듈은 윈도우 창을 생성하고 패킷 캡춰 모듈로부터 넘겨받은 데이터를 저장하고 그래프로 표시해 주는 역할을 하며, 라운드로빈 테스트 패키지는 총 3개의 모듈로 구성이 되며 각각 데이터 생성기 모듈, 가상 서버 모듈, 데이터 수신기 모듈이라 명한다. 데이터 생성기 모듈은 랜덤 값의 데이터를 생성하는 것으로 가상 서버 모듈에 전송하는 역할을 한다 가상서버 모듈은 데이터 생성기로부터 받은 데이터를 라운드 로빈 알고리즘에 의해 분산시켜 각각 데이터 수신기에 전송하는 역할을 한다. 데이터 수신기 모듈은 가상서버 모듈에서 들어오는 데이터를 파일로 저장하여 비교 분석하는 역할을 한다.

추후 연구 과제로는 캐쉬 알고리즘과 라운드 로빈 알고리즘의 연동과 이에 따르는 보안적인 문제의 해결 방안을 들 수 있겠다.

참고문헌

[1] W.R Stevens, TCP/IP Illustarted Volume 1., Addison-Wesley, New york. pp.25-45, 1996.

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

[3] P.Gupta, S.Lin, and N.Mckeown, "Routing lookups in hard ware at Memory Access

Speeds," Proceedings of the Conference on Computer Communications(IEEE INFOCOM), pp.1241-1248. 1998.
[4] Web Server Farm Performance. White Paper, Arrowpoint Communications, p 278-284, 1998.
[5] T. Schroeder, S.Goddard, and B. Ramamurthy, "Scalable Web Server Clustering Technologies." IEEE Network, pp.38-45, 2000.
[6] 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.
[7] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, Fast and Scalable Layer Four Switching, SIGCOMM, pp.115-123 1998
[8] Seok-soo Kim, "Designing Linux Cluster Content-aware Load Balancer",II International Conference "Science and Technology in XXI, pp.166-167, 2003
[9] D.A Maltz, and P.Bhagwat, TCP Splicing for Application Layer Proxy Performance, Joumal of High Speed Networks. pp.225-240, 1999
[10] M.Aron, D.Sanders, and P.Druschel, Scalable Content-Aware Distribution in Cluster-based Network Servers. Proceedings of the 2000 USEMIX Technical Conference, 2000.
[11] W. Zhang and et al., Linux virtual server project ,.org, pp.125-140, 1998.
[12] W. Zhang, Linux Virtual Server for Scalable Network Services, Linux virtural server project,. org, pp.112-124, 1999.

저자소개

박길철(Gil-Cheol Park)

1985년 2월 숭실대학교 대학원 공학석사
1994년 3월~1998년 2월 성균관대학교 대학원 공학박사
1985년 7월~1990년 10월 삼성종합 기술원
1992년 1월~1996년2월 대교컴퓨터
1996년 3월~1998년 8월 한서대학교 컴퓨터과학과 조교수
1998년 9월~현재 한남대학교 정보통신멀티미디어학부 부교수
※관심분야 : 유무선 통합 모바일 시스템, 멀티미디어 실시간 통신, 멀티미디어 컨텐트

성경(Kyung Sung)

1993년 8월 경희대학교 대학원 공학석사
2003년 2월 한남대학교 대학원 공학박사
1988년 8월~1990년 12월 국제종합기계(주)
1990년 12월~1994년 2월 배재대학교
1994년 3월~2004년 2월 동해대학교 컴퓨터공학과 부교수
2004년 3월~현재 목원대학교 컴퓨터교육과
※관심분야 : 정보관리(보호), 신경망, 네트워크

김석수(Seok-Soo Kim)

1991년 2월 성균관대학교 대학원 공학석사
1991년 2월~1996년 5월 정풍물산(주) 중앙연구소 주임연구원
1997년 4월~1998년 1월 (주)한국탑웨어 책임연구원
2002년 2월 성균관대학교 대학원 공학박사
1998년 3월~2000년 2월 경남도립거창전문대학 교수
2000년 3월~2003년 2월 동양대학교 컴퓨터공학부 교수
2003년 3월~현재 한남대학교 정보통신멀티미디어학부 교수
※관심분야 : 멀티미디어, 정보통신, 웹솔루션, 정보보호, 원격교육 플랫폼 및 컨텐츠