리눅스기반 웹서버 과부하방지 Switch 구현

윤호태°, 송재원
경북대학교 정보통신학과
htyoon@inc.knu.ac.kr°, jwsong@ee.kyungpook.ac.kr

Implementation of Linux-Based Web Server Surge Protection Switch

Ho-Tae Yoon°, Jae-Won Song
Dept. of Information & Communication, Kyungpook National University

요 약

컴퓨터 네트워크 및 인터넷의 발전으로 인터넷 사용자 및 웹서버의 숫자는 기하급수적으로 증가하고 있다. 그러나 컴퓨터 네트워크의 발달에도 불구하고 웹서버에 접속하는 사용자 수가 많아짐에 따라 웹서버에 병목현상이 발생하고 있다. 특히 수강신청과 같은 동일 시간대에 한 웹서버에 접속하는 사용자 수가 많게 되면 웹서버는 과부하로 인하여 작동을 멈추거나 속도가 느려진다. 본 논문에서는 이러한 문제점을 해결하고 웹서버의 지속적인 서비스가 가능하도록 web traffic을 조절할 수 있는 웹서버 과부하방지 Switch를 구현하였다.

1. 서 론

인터넷이 급속히 성장하고 인터넷이 우리 생활의 일부분으로 서서히 차지하는 비중이 높아지면서 인터넷 traffic은 아주 놀라울 정도로 빠르게 증가하고 있다. 서버의 처리량도 이와 함께 빠르게 증가하고 있고 아주 유명한 웹서버의 경우는 짧은 시간에도 과부하가 걸리기 쉽다. 특히 수강신청과 같은 동일 시간대에 접속하는 사용자 수가 많게 되면 웹서버는 과부하로 인하여 작동을 멈추거나 속도가 느려진다.

그래서 많은 서버 관리자들은 그들 서버의 성능 상의 병목 문제를 해결하기 위해 서버의 액세스 용량을 늘리지만 얼마 지나지 않아 쉽게 과부하가 되어 버린다. 요즘은 점점 더 많은 회사들이 그들의 사업을 인터넷상으로 옮기고 있어 서버가 제공하는 서비스의 어떠한 중단도 사업상 손실을 의미하게 되었다. 이러한 서버들의 높은 가용성은 점점 더 중요해지고 있다. 그러므로 높은 가용성을 가진 서버에 대한 요구가 절실히 증가하고 있다. 서버를 더 높은 성능의 서버로 개선하는 단일 서버 솔루션은 이 필요 조건을 맞추기에 한계가 있다. 개선 작업이 복잡하고 원래의 기계가 낭비될 수도 있다. 만일 요구가 증가된다면 오래 지나지 않아 과부하가 되어 더 높은 성능의 서버로 다시 개선해 주어야 한다. 개선해야 할 목표가 높아지면 높아질수록 비용이 더욱더 든다.

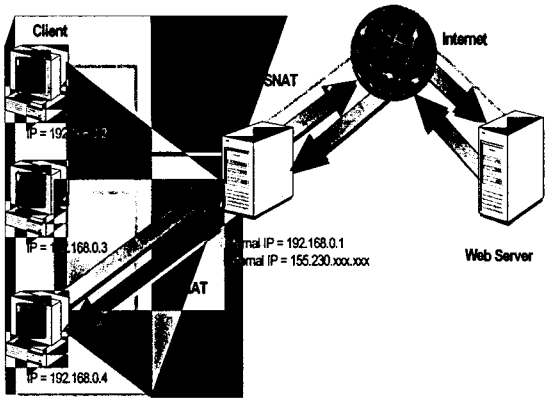
본 논문에서는 웹서버가 서비스 할 수 있는 connection 수를 switch에서 조절 해 줌으로써 웹서버에서의 과부하로 인한 작동이 멈추거나 속도가 느려지는 문제점들을 해결할 수 있도록 하였다. 그리고 웹서버의 connection 수가 초과하였을 경우에 웹서버에 connection할 수 없다는 error 메세지를 웹 페이지에 보여주는 것이 아니라, switch에 또 하나의 웹서버를 구축해서 잠시 뒤에 다시 접속하라는 안내 메시지를 보여줌으로써 사용자에게 신뢰를 높일 수 있다.

본 논문은 다음과 같이 구성하였다. 2장에서는 웹서버 과부하방지 Switch 구현을 위한 관련 연구로써 내부 네트워크 구축 및 네트워크를 통해 들어오고 나가는 패킷들을 capture하는 libpcap 그리고, connection을 재사용하는 HTTP1.1에 대해 기술하고, 3장에서는 system이 switch 역할을 할 수 있게 switch를 구현하는 것에 대해 알아보고, 4장에서는 본 논문에 대한 결론을 내렸다.

2. 관련 연구

2.1 내부 네트워크 구축

Switch 기능을 하기 위해 먼저 내부 네트워크를 구축하였다. 정상적인 packet은 그것의 시작점에서부터 목적지까지 많은 다른 링크를 경유하게 된다. 이런 packet이 NAT(Network Address Translation)를 사용하는 곳을 지나게 되면 그 packet의 IP address가 변화되게 된다[1]. [그림 1]에서는 NAT가 어떻게 구성되는지 잘 보여주고 있다.



[그림 1] NAT(Network Address Translation)

NAT에는 SNAT(Source NAT)와 DNAT(Destination NAT)로 구분된다.

SNAT는 패킷의 시작점 주소를 변경할 때이다. 즉, 나가는 접속을 변경하는 것이다. Source NAT는 언제나 routing후에 이루어지고 packet이 바깥으로 나가기 직전에 이루어진다.

DNAT는 패킷의 목적지 주소를 변경할 때이다. 즉, 접속이 어디로 향하는지를 변경할 때이다. Destination NAT은 언제나 routing이전에 이루어지고 packet이 처음으로 들어왔을 때 이루어진다.

이러한 NAT 기능을 이용한 내부 네트워크는 private IP를 사용하게 되고 kernel은 NAT 기능을 지원할 수 있게 kernel을 compile 해준다[2].

내부 네트워크를 NAT를 이용한 private IP를 사용해서 구축함으로써 주소 변환 과정 중에 실제의 private IP를 숨겨주는 Firewall 기능도 포함되어 있어 보안에도 도움을 준다[3].

2.2 Packet Capture

패킷을 캡쳐하기 위한 도구로는 각 운영체제별로 다양한 도구가 있다. 하지만 이 모든 도구들을 수용하는 Portable한 API가 있는데 이것이 바로 libpcap이다[4]. 라이브러리 사용자는 운영체제의 각기 다른 *datalink*로의 접근 방법에 상관없이 libpcap을 이용하여 패킷을 캡쳐할 수 있다. 본 논문에서는 libpcap를 이용하여 switch를 통과하는 패킷들을 제어할 수 있도록 한다. 패킷 캡쳐를 위해 사용한 함수들은 다음과 같다.

- pcap_lookupdev() 네트웍 디바이스를 가져오는 함수이다.
- pcap_open_live() 실제 기기를 열어주는 기능을 한다.
- pcap_lookupnet() 열려진 패킷 캡쳐 디바이스에 네트웍 주소와 서브넷 마스크를 넘겨준다.
- pcap_compile() 정해진 필터룰에 의해 필터 프로그램을 컴파일한다.
- pcap_setfilter() 컴파일한 필터 프로그램을 지정해 주는데 사용된다.
- pcap_loop() 실제 패킷을 잡아서 실행할 함수를 지정해 주는 함수이다.

2.3 HTTP1.1

현재 Browser나 Server Software를 사용할 수 있는 HTTP로는 1.0과 1.1의 두 버전이 있으며, 1.1은 1.0의 모집합이다. HTTP 1.0에서는 Connection의 재사용은 불가능하며 아이템을 서버로부터 취득할 때마다 Connection을 시도하고, 취득 후에는 중단한다. 일반적인 웹사이트에서 HTTP1.0 client는 한 페이지를 다운로드 하기 위해 약 50번 정도의 connection을 열고 닫게 된다. 각 connection이 3-way handshaking을 한다고 볼 때 이러한 과정으로 인해 얼마나 사이트가 느려지게 되는지는 자명하다. HTTP1.1은 이러한 문제를 해결하고자 client와 server간에 한번 설정한 Connection을 다른 아이템을 취득할 때 재사용 함으로써 이 문제를 해결하고 있다[5][6].

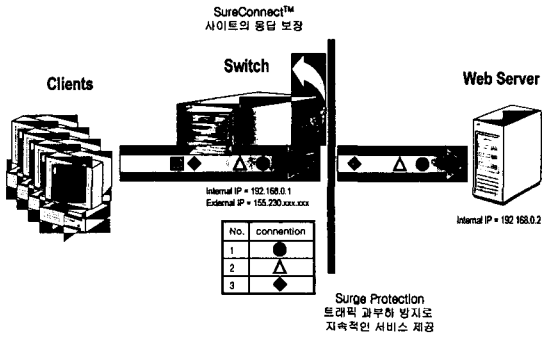
3. 웹서버 과부하방지 Switch 구현

3.1 구현 환경

웹서버 과부하방지 Switch를 구현하기 위해 운영체제로 linux kernel-2.4.19를 사용하였고, CPU는 Pentium4 1300MHz, 메모리는 256MB, LAN카드는 3COM 3C905(eth0), RealTek RTL-8029(eth1)를 사용하였다. 그리고 웹서버의 운영체제는 linux kernel-2.4.9를 사용하였고, CPU는 Pentium MMX 166MHz, 메모리는 78MB, LAN카드는 3COM 3C509(eth0)를 사용하였다. 웹서비스는 apache_1.3.20를 웹서버와 Switch에 설치하였다.

3.2 Switch 구현

[그림 2]는 웹서버 과부하방지 Switch가 어떻게 작동하는지를 보여준다.



[그림 2] 웹서버 과부하방지 Switch의 전체 구성도

인터넷을 통해 웹서버에 서비스를 요청하기 위한 모든 패킷들은 웹서버 과부하방지 Switch를 지나게 된다. Switch에서는 어떤 패킷이 어디에서 접속을 요청하는지를 libpcap를 이용하여 실시간으로 모니터링하게 된다. Client가 웹서비스를 요청하게 되면, TCP 3-way handshake 과정을 거치면서 connection이 이루어지게 되는데, connection이 이루어지는 첫 번째 과정으로 SYN 패킷을 server측에 보내게 된다. Switch에서는 SYN 패킷이 connection을 이루기 위한 첫 번째 패킷이기 때문에 이 패킷의 TCP와 IP 헤더정보들을 메모리를 할당해서 저장한다. 이 정보를 가지고 있음으로 해서 Switch에서는 client에서 server로 connection이 얼마나 이루어졌는지 알 수 있다. Switch에서는 server의 동시 처리할 수 있는 성능 정보를 가지고 그 성능의 한계 connection을 초과 할 경우 더 이상 client에서 server로의 connection을 허용하지 않게 한다. 그리고, connection이 줄어들어 connection이 허용 가능하게 되면 server로 연결을 열어주는 switch 역할을 수행하게 된다.

한편, 연결 종료는 메모리 상에서 연결을 설정할 때처럼 순차적으로 일어나는 것이 아니기 때문에 메모리 상에서 연결 순서를 효과적으로 관리하기 위해 저장할 때 연결리스트 형태로 패킷들을 저장하였다.

[표 1] packet 처리 알고리즘

```
void packet_analysis( )   //패킷이 잡힐 때마다 호출되는 함수
{
   if (SYN 패킷인 경우) {
      TCP/IP의 헤드 정보와 그 패킷의 순번 정보를
      메모리에 연결리스트 형태로 저장
      if (connection이 초과한 경우) {
         패킷이 통과 하지 못하게 Filter Rule 설정
      }
   }
   if (RST 패킷의 경우) {
      연결리스트에서 해당 패킷 정보를 삭제
      if (connection이 0만일 경우) {
         패킷이 통과 하도록 Filter Rule 설정
      }
   }
}
```

[표 1]에서 packet_analysis()함수는 pcap_loop()에 의해 패킷을 잡을 때마다 불려지는 함수로써, 패킷들이 잡힐 때마다 그 패킷이 어떤 패킷인지 분석하고, 그 패킷의 종류에 맞게 처리해 주는 알고리즘을 보여주고 있다.

[표 2] packet 통과 Filter Rule 설정

```
modprobe ipt_MASQUERADE
iptables -F; iptables -t nat -F; iptables -t mangle -F
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to
155.230.xxx.xxx
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0
-j DNAT --to 192.168.0.2
echo 1 > /proc/sys/net/ipv4/ip_forward
```

[표 2]에서는 웹서버의 허용 연결수가 초과하지 않았을 경우의 packet filter rule로써 80번 포트로 들어오는 패킷은 내부 네트워크의 주소인 192.168.0.2로 DNAT된다.

[표 3] packet 통과 불가 Filter Rule 설정

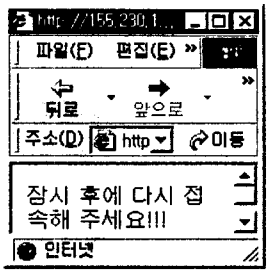
```
iptables -A INPUT -d 155.230.xxx.xxx -p tcp -m tcp --
dport 80 -j ACCEPT
```

[표 3]에서는 웹서버의 허용 연결수가 초과한 경우 웹서버로 패킷을 통과시키지 않고, Switch에서 패킷을 받아들이는 packet filter rule 이다. 이 packet filter rule을 적용하면 기본적인 Firewall 규칙도 설정할 수 있다.

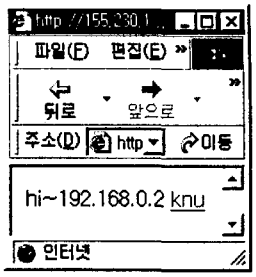
[표 4] connection 설정 및 종료

```
device = eth0
0 --> NULL                                  증가
0 --> 1 --> NULL
0 --> 1 --> 2 --> NULL
0 --> 1 --> 2 --> 3 --> NULL                감소
0 --> 1 --> 2 --> NULL
0 --> 1 --> NULL                            증가
0 --> 1 --> 2 --> NULL
0 --> 1 --> 2 --> 3 --> NULL
```

[표 4]에서는 사용자들이 웹서버로 연결을 설정하고 종료하는 과정을 connection의 변화로 보여주고 있다.



[그림 3] Switch Off



[그림 4] Switch On

[그림 3]과 [그림 4]은 connection 수에 따라 Switch의 On/Off에 따른 결과 화면을 보여주고 있다. [그림 3]은 connection이 초과한 경우인데 Switch에서 잠시 뒤에 다시 접속하라는 안내 메시지를 보여줌으로써 사용자에게 신뢰를 줄 수 있고, 대체 내용을 제공할 경우 광고 효과도 기대할 수 있다. [그림 4]는 connection이 초과하지 않은 경우로 웹서버의 서비스를 제공하고 있다.

4. 결론

동일 시간대에 많은 사용자가 한 웹서버에 접속을 하게 되면 웹서버는 과부하로 인하여 작동을 멈추거나 속도가 느려진다. 본 논문에서는 이런 문제점을 해결하고자 웹서버로 통하는 통로에 웹서버의 connection수를 조절할 수 있는 웹서버 과부하방지 Switch를 둠으로써 서버의 다운 방지 및 지속적인 웹서버의 서비스를 보장해 준다. 그리고, 최대 허용connection을 초과할 경우는 Switch에서 잠시 뒤에 다시 접속하라는 안내 메시지 제공을 통해 사용자에게 신뢰를 줄 수 있고 웹 페이지에 대체 내용을 제공 할 경우 광고 효과도 기대할 수 있는 장점을 가진 웹서버 과부하방지 Switch를 구현해 보았다.

참고문헌

[1] Rusty Russell, "Linux 2.4 NAT HOWTO v1.0", 2000.

[2] Ambrose Au, "Linux IP Masquerade mini HOWTO v1.20", 1997.

[3] Rusty Russell, "Linux 2.4 Packet Filtering HOWTO", 2001

[4] 노광민, "Libpcap 사용하기 v0.4", 2002.

[5] High Volume Website Service Team, "웹 컨텐츠의 튜닝", 2002.

[6] NetScaler White Paper, "Cost-Effective, Continuous, Secure Delivery of Business-Critical Applications", July 2003