# 웹서버를 위한 네트워크 트래픽 모니터링 도구 설계

최영한[0], 임상석, 이 철, 박규호
한국과학기술원 전자전산학과 전기전자전공
(yhchoi, sslim, chullee, kpark)@core.kaist.ac.kr

# Design of Network Traffic Monitoring Tool for Web Server

Young-Han Choi[0], Sang-Seok Lim, Chul Lee, Kyu-Ho Park
EECS Dept. Korea Advanced Institute of Science and Technology

요 약

본 논문에서는 네트워크 통신으로 생성된 데이터를 분석하여 서버의 부하를 예측함으로 서버의 성능을 향상 시킬 수 있는 모니터링 툴을 설계하였다. 현재 리눅스에서 제공되는 서버 부하 정보는 시스템의 총괄적인 정보에 국한하고 있다. 본 논문에서 제한하는 새로운 부하 정보는 per-connection에 관한 것이다. 서버가 per-connection에 대한 정보를 알 수 있다면 각각의 클라이언트에 대한 정보를 알 수 있어 차별적인 서비스를 제공하는 것과 함께 모든 클라이언트에 대한 균등한 대역폭을 보장할 수 있다. 그래서 본 논문은 기존의 모니터링 툴이 제공하지 않는 per-connection의 데이터를 얻어 분석을 할 수 있는 CoreMon이라는 툴을 설계하였다.

## 1. 서 론

최근 인터넷 사용자의 급격한 증가는 네트워크 트래픽의 범람을 야기시켰다. 기술적인 진보로 인하여 네트워크 대역폭은 크게 확장 되었지만, 네트워크 서버의 성능 향상은 그에 미치지 못하고 있다.

그리고 일부 클라이언트는 서버에 동시 다발적으로 접속함으로써 서비스의 질을 저하시키고 있으며, 최악의 경우에는 서버의 다운을 유발시키고 있다.

그러기에 서비스의 요청에 대한 패턴과 각각의 클라이언트에 대한 접속 정보를 알 수 있다면 여러가지 이유로 서버의 성능을 향상 시킬 수 있을 것이다.

첫째로 서버에 대한 과부하를 예측 할 수 있어 서버의 운용을 효율적으로 할 수 있다. 두번째로 QoS 측면에서 공평한 스케줄링이 가능해 차별적인 서비스의 지원과 모든 클라이언트에 대한 동등한 대역폭을 제공함으로 다수의 클라이언트를 만족 시킬 수 있다.

서버의 네트워크 트래픽 정보를 제공하기 위해 많은 모니터링 툴이 제공되고 있는데 그 중에 대표적으로 사용되는 MON이라는 툴이 있다[7]. MON은 네트워크 트래픽에 대한 종합적인 정보를 제공한다. 그러나 perl로 개발되어 있어서 커널에서 직접적으로 얻어내야만 하는 정보들은 보여주지 못하고 있다. 이러한 정보 중에서 특히 per-connection에 대한 정보를 얻을 수 없어 개별적인 클라이언트에 대한 네트워크 트래픽을 측정하지는 못한다.

본 논문에서는 서버에 접속하는 개별적인 클라이언트의 네트워크 트래픽 정보를 측정할 수 있으며 이와 함께 종합적인 네트워크 트래픽을 측정할 수 있는 툴인 CoreMon을 리눅스 플랫폼에서 설계하였다.

CoreMon을 사용하게 되면 서버에 접속하는 클라이언트의 개별적이고 종합적인 정보를 수집하여 서버의 부하에 대한 분석을 자세히 할 수 있어 서비스의 질을 높을 수 있다. 그래서 포털 업체나 서비스를 제공하는 회사와 공공기관에서 유용하게 사용을 할 수 있을 것이다.

2장에서는 CoreMon의 전체 구조를 나타내었다. 3장에선 /proc 파일 시스템에서 전체 네트워크 트래픽 데이터를 읽어와서 분석하여 사용자에게 정보를 보여주는 부분을 설명하였다. 그리고 4장에서는 커널에 모듈을 삽입하여 per-connection 데이터를 읽어와 처리하는 부분을 소개하였다.

## 2. CoreMon의 구조

CoreMon은 구조상 두 부분으로 나눌 수 있다.

① 데이터를 처리하여 사용자에게 보여주는 부분
- /proc 디렉토리에서 데이터를 수집하여 CoreMon으로 처리하여 사용자에게 서버의 부하에 대한 정보를 보여준다.

② per-connection의 데이터를 수집하기 위한 모듈 부분
- per-conncetion의 데이터를 위해 커널에 모듈을 삽입한다.
- 모듈에서 읽은 데이터를 /proc 디렉토리에 파일로 저장한다.
- user level에서 /proc 밑의 데이터를 읽은 후 필요한 정보를 얻기 위해 분석을 한다.

그림1에서 part 1은 /proc 디렉토리에서 데이터를 읽어와 필요한 정보를 얻기 위해서 분석하는 부분이고 part 2는 per-connection에 관한 데이터를 읽어 오기 위해 새로운 모듈을 설계하여 커널에 삽입 한 부분이다.

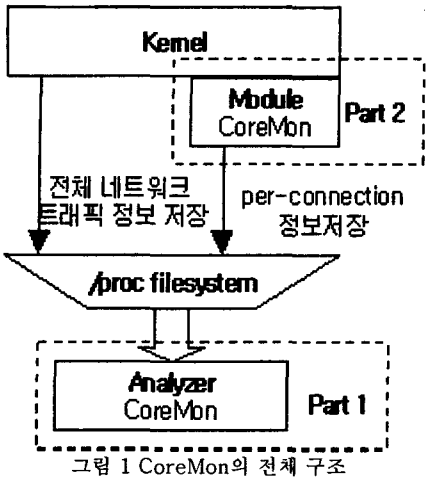


그림 1 CoreMon의 전체 구조

CoreMon에서 읽어와 사용자에게 보여주는 정보는 위의 구조와 같이 두 부분으로 나눌 수 있다.

첫째는 네트워크 트래픽에 대한 서버의 전체적인 부하 정보이고 두번째는 각 클라이언트에 대한 정보에 해당하는 per-connection 관한 정보이다.

### 3. 전체 네트워크 트래픽 데이터 수집

리눅스는 /proc 디렉토리에 실시간으로 시스템 정보나 커널 파라미터에 대한 정보를 제공한다. 제공하는 여러가지 정보 중 CoreMon은 /proc/net 밑에 있는 dev, snmp, stat 파일을 이용하였다.

stat 파일에서는 시스템에 대한 전반적인 데이터를, dev 파일에서는 패킷에 대한 데이터를, snmp 파일에서는 TCP, UDP로 전송되는 데이터를 얻을 수 있으며 자세한 파라미터는 표1에 나타내었다.

| Stat | |
|---|---|
| Disk | 총 disk I/O |
| Page | in & out page의 수 |
| ctxt | Context switch 되는 수 |
| **Dev** | |
| Packets | 들어오고 나가는 패킷수 |
| Bytes | 들어오고 나가는 바이트 크기 |
| Errs | 들어오고 나가는 패킷의 에러 수 |
| Drop | 들어오고 나갈 때 드롭되는 패킷의 수 |
| **Snmp** | |
| InSegs | 들어오는 세그먼트 수 |
| OutSegs | 나가는 세크먼트 수 |
| InErrs | 세그먼트 에러 수 |
| RetransSegs | 재전송되는 세그먼트수 |
| InDatagrams | 들어오는 데이터그램의 수 |
| OutDatagrams | 나가는 데이터그램의 수 |
| InErrors | 데이터 그램의 에러 수 |

표 1 /proc에서 축출하는 데이터

표1 에서 보는 것과 같이 /proc에서 얻을 수 있는 데이터를 실시간으로 읽어와 CoreMon으로 이동시킨다.

그리고 제공되는 데이터를 이용해서 전체 클라이언트에 대한 서버의 부하를 일정한 간격으로 수집하여 CoreMon으로 데이터를 분석하였으며 그 결과는 그림 2와 같다.

| | Packet(byte/s) | | Packet(no/s) | | Error | | Drop | | ctxt(no/s) |
|---|---|---|---|---|---|---|---|---|---|
| time | Tr | Re | Tr | Re | Tr | Re | Tr | Re | |
| 13:42:20 | 318.60 | 363.80 | 3.40 | 4.00 | 0.00 | 0.00 | 0.00 | 0.00 | 16.00 |
| 13:42:25 | 269.60 | 270.40 | 2.80 | 3.00 | 0.00 | 0.00 | 0.00 | 0.00 | 15.20 |
| 13:42:30 | 1767.20 | 647.40 | 4.60 | 6.20 | 0.00 | 0.00 | 0.00 | 0.00 | 22.40 |
| 13:42:35 | 382.80 | 440.00 | 3.00 | 4.40 | 0.00 | 0.00 | 0.00 | 0.00 | 16.40 |
| 13:42:40 | 382.80 | 339.00 | 3.00 | 3.60 | 0.00 | 0.00 | 0.00 | 0.00 | 16.00 |
| 13:42:45 | 318.40 | 339.00 | 3.20 | 3.60 | 0.00 | 0.00 | 0.00 | 0.00 | 16.00 |
| 13:42:50 | 323.60 | 405.40 | 3.40 | 4.60 | 0.00 | 0.00 | 0.00 | 0.00 | 16.40 |
| 13:42:55 | 361.20 | 398.20 | 3.80 | 4.40 | 0.00 | 0.00 | 0.00 | 0.00 | 16.80 |
| 13:43:00 | 302.80 | 375.60 | 3.00 | 3.60 | 0.00 | 0.00 | 0.00 | 0.00 | 16.60 |

| | TCP(segment/s) | | | UDP(datagram/s) | | |
|---|---|---|---|---|---|---|
| time | Tr | Re | Err | Tr | Re | Err |
| 13:42:20 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:42:25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 13:42:30 | 2.00 | 3.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 13:42:35 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:42:40 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:42:45 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:42:50 | 1.40 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:42:55 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |
| 13:43:00 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 |

그림 2 서버 부하에 대한 종합적인 정보 출력

### 4. 각각의 connection의 데이터 수집

기존의 proc 파일 시스템의 데이터를 분석함으로 전체 클라이언트에 대한 서버의 부하에 대한 정보를 얻을 수 있으나 각 클라이언트에 대한 per-connection의 정보를 알 수 없다.

그래서 필요한 per-connection의 정보를 얻기 위해 새로운 모듈을 설계하여 커널에 삽입하기로 한다.

#### 4.1 커널에서 per-connection 데이터 수집

커널에서는 connection을 맺고 난 후 데이터가 이동하기 위해서 *VFS, BSD socket, inet socket, TCP, IP, Device* 계층을 거치게 되는데 전송을 할 때는 데이터가 계층 밑으로 이동을 하면서 헤더를 첨가하여 보내게 되고, 수신을 할 때는 각 층에서 그 층에 해당하는 헤더를 제거하면서 위의 계층으로 보내게 된다.

CoreMon에서 보여주는 connection에 대한 정보로는 per-connetion에 해당하는 세그먼트 전송률과 데이터 전송률이다.

각 Connection의 데이터 전송률을 알기 위해서 우선 전송량을 알아야 되는데 그것을 알아내기 위한 정보가 포함되어 있는 구조체는 sk_buff와 sock이다.

```
struct sock {
    ...
    __u32       daddr;    // Foreign IPv4 addr
    __u32       rcv_saddr;// Bound local IPv4 addr
    __u16       dport;    // Destination port
    unsigned    num;      // Local port
    ...
     struct sk_buff_head  receive_queue;
                    // Incomming packets
    struct sk_buff_head write_queue;
                    //Packet sending queue
    ...
};
```

```
struct sk_buff {
        ...
        unsigned int len;  // length of actual data
        ...
};
```

그림 3 struct sock & sk_buff

그림 4 에서 볼 수 있듯이 CoreMon이 필요한 정보는 connection에 관한 것이기 때문에 TCP 계층에서 호출되는 tcp_send_skb(), tcp_recvmsg() 함수에서 필요한 데이터 값인 sock, sk_buff 값을 통해 알아낼 수 있다.

struct sock 에서는 send IP address & port number, receive IP address & port number를 알 수 있어 접속을 하는 각각의 connection을 구별 할 수 있으며 read &write queue로는 queue의 길이를 알 수 있다.

struct sk_buff 에서는 len 변수로 이동한 데이터량을 얻을 수 있다.

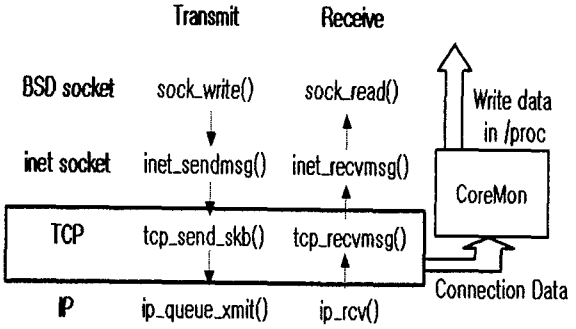


그림 4 모듈 삽입 구조

세그먼트 전송률에 필요한 세그먼트 수는 그림 4에서 보듯이 TCP layer 에서 호출되는 함수인 tcp_send_skb()와 tcp_recvmsg()가 호출된 횟수로 측정을 할 수 있다.

### 4.2 per-connection 데이터를 /proc에 저장

커널에서 읽어 온 데이터를 /proc 디렉토리에 파일로 저장을 한 후 CoreMon은 그것을 이용하여 데이터를 분석한다.

/proc에 저장되는 파일은 디스크에 저장되는 것이 아니라 메모리에 저장되는 것이므로 일반적인 파일 시스템과 다른 파일 시스템을 필요로 하게 된다. /proc 파일 시스템을 위해 새로운 inode가 요구되며 구조체 proc_dir_entry가 필요하다.

```
struct proc_dir_entry {
   low_ino, namelen, name, mode, nlinks, uid, gid,
   size, ops, read_proc
};
```

그림 5 struct proc_dir_entry

proc_dir_entry을 이용하여 proc system에 등록을 해야 한다. 등록을 위해선 proc_register() 함수가 호출된다.

/proc에 파일로 저장을 한 후 일정한 시간마다 그 데이터를 읽어 와서 CoreMon은 per-connection에 해당하는 정보를 얻을 수 있으며 클라이언트는 이 정보를 바탕으로 차별적인 서비스를 제공할 수 있는 것이다.

### 5. 결론 및 추후과제

본 논문에서는 서버에서 종합적인 것만 뿐만 아니라 per-connection에 해당하는 네트워크 트래픽을 측정할 수 있는 모니터링 툴을 설계하였다. 기존에 나와 있는 것은 종합적인 네트워크 트래픽의 정보만 제공하기에 각 클라이언트에 대한 트래픽 정보를 알 수 없었다. 그러기에 차별적인 서비스를 제공하는 것과 모든 클라이언트에게 균등하게 대역폭을 제공할 수 없었다. 그러기에 본 논문에서 설계한 CoreMon을 이용하여 종합적인 네트워크 부하를 예측할 수 있을 뿐만 아니라 개별적인 클라이언트의 정보를 알아내어 서버가 균등하게 서비스를 제공 할 수 있게 하였다.

앞으로의 연구 과제로는 여러가지 환경을 고려하여 세부적인 파라미터를 CoreMon에 추가 시킴으로써 보다 많은 정보를 서버에 제공하여 최적의 서비스를 제공할 수 있도록 하는 것이다.

### 참고문헌

[1] 조유근, 최종무, 홍지만 "리눅스 매니아를 위한 커널 프로그래밍", pages 196-207

[2] Remy Card, Eric Dumas, Franck Mevel "the Linux Kernel book", pages 242-247

[3] W. Richard Stevens "TCP/IP Illustrated, vol 1", pages 223-228

[4] Glenn Herrin, "Linux IP Networking", pages 22-31, 77-87, May 2000

[5] Jussara Almeida, Virgilio Almeida, David J Yates, "Measuring the Behavior of a World-Wide Web Server", October 1996

[6] Linux Virtual Server, http://www.linuxvirtualserver.org

[7] MON, http://www.kernel.org/software/mon/