Linux kernel 2.4 네트웍 트래픽 분산

유성태

alloying (at) kldp.org

회사에서 인터넷 사용에 대한 요구는 많으나 비싼 전용선 가격 때문에 증속을 하기가 쉽기 않은 것이 현실이다. 이때 ADSL을 도입하여 전용선과 함께 사용한다면 매우 효율적일 것이다. 본 문서에서는 리눅스 커널 2.4에서 전용선과 ADSL을 함께 사용하여 네트웍 트래픽을 분산 시키는 방법을 설명한다. 그리고 업로드와 다운로드 속도가 다른 ADSL의 특성을 고려할 때, 이 문서에서 설명하는 네트웍 트래픽의 분산은 다운로드가 많은 확경에 적합함을 미리 밝혀 둔다. 전용선과 ADSL 뿐만 아니라 ADSL 두 라인, 또는 두 개의 전용선 환경 등에서도 적용할 수 있음은 물론이다.

고친 과정

고침 0.12001-12-31고친이 styoo

최초 작성

고침 0.22002-01-19고친이 styoo

백업 라인에 대한 내용 추가 - 정헌학님의 질문 port 기반의 트래픽 분산에서 틀린 내용 수정 - 김성철님께 감사

고침 0.32002-03-18고친이 styoo

라우팅 테이블 생성 스크립트 예제 추가 참고자료 링크 수정

1. 서문

리눅스 커널 2.4의 발표 소식을 접한 후 그 안에 어떤 새로운 기능이 들어 있을지 매우 궁금하였다. 특히 네트웍과 관련하여 평소에 원하던 기능이 꼭 들어 있으리라 여겨졌다. 그러나 정작 커널 2.4.9 버전이 발표된 뒤에야 비로소 필요한 정보를 찾기 시작하였다. 이제 그 적용 결과를 문서로 남겨 둘 필요를 느껴서 이왕이면 kldp의 문서 형태로 적성하여 여러분과 공유하고자 한다.

1.1. 저작권 정보

은 자유 소프트웨어 재단에서 발행한 이후 판의 규정에 따르며 저작권에 대한 본 사항이 명시되는 한 어떠한 정보 매체에 의한 본문의 전재나 발췌도 무상으로 허용됩니다.

1.2. 책임의 한계

본 저자는 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 본 문서에서 내포하고 있는 정보들 및 예제들은 여러분이 알아서 활용하십시오. 비록 최선을 다 했으나 이 문서는 틀린 점이나 오류가 있을 수도 있습니다. 만약 여러분이 틀린 점을 발견했다면 꼭 저에게 알려 주시기 바랍니다.

1.3. 감사의 글

오늘도 리눅스 커널을 만지고 있을 수 많은 해커들에게 감사를 보내며 그 외에 HOWTO 문서 작성에 수고를 하고 있는 수 많은 사람들에게도 감사를 드립니다.

1.4. 피드백

이 문서에 대한 발전적인 제안이나 수정사항, 문제점 등에 대한 피드백은 언제든지 환영합니다. `<alloying (at) kldp.org>`로 메일을 보내 주십시오.

2. 네트웍 트래픽 분산의 개요

A라는 컴퓨터에서 인터넷으로 데이타를 내보낼 수 있는 문(gateway)은 하나밖에 없는 경우가 대부분이다. 그러나 A라는 컴퓨터에서 데이타를 내보낼 수 있는 문(gateway)이 여러개 있다고 가정해 보자. 이때 1번 문을 통해서 나간 데이타에 대한 응답은 1번 문을 통해서 들어오고, 2번 문을 통해서 나간 데이타에 대한 응답은 2번 문을 통해서 들어오게 되어 있다. 특별한 조작을 하지 않는 한, 1번 문을 통해서 나간 데이타가 다른 문으로 들어올 가능성은 없다. 이를 네트웍 트래픽 분산과 관련지어 생각하면, 나가는 데이타를 적절히 분류하여 여러개의 문으로 분산시켜 내보내면 각 데이타에 대한 응답은 자연스럽게 여러개의 문으로 분산되어 들어올 것이라는 것이다. 즉, 내보내는 데이타만 적절히 분산시켜 내보내면 원하는 네트웍 트래픽 분산을 달성할 수 있다는 것이다.

네트웍을 통해서 오가는 데이타는 **packet**이라는 조각으로 나뉘어서 오간다. 이하에서는 데이타 대신에 패킷이라는 용어를 사용하도록 하겠다.

내보내는 패킷을 분류하는 방법은 크게 두 가지로 나눌 수 있다. 첫째는 내보낼 패킷이 어느 컴퓨터 또는 어느 네트웍으로부터 나오느냐에 따라 나가는 문을 배정할 수 있다. 두번째는 내보낼 패킷이 웹을 사용하는지, ftp를 사용하는지, telnet을 사용하는지 등에 따라, 즉 port에 따라 나가는 문을 배정할 수 있다. 굳이 방법을 한 가지 더 추가한다면 첫번째와 두번째 방법을

섞어서 사용하는 것이다. 즉, A라는 컴퓨터에서 나오는 www 패킷은 2번 문을 통하게 한다라는 식이다.

필자는 현재 E1 전용선 1개와 ADSL 2개를 사용하고 있으며, 첫 번째 방법을 사용하여 네트웍 트래픽 분산을 비교적 만족스럽게 하고 있다.

3. 네트웍 환경

네트웍 트래픽 분산을 설명할 가상의 네트웍 환경을 꾸며보자.

```
   내부 네트웍                                +-----------------+
 +------------+                              |      웹서버      |
 |   Net-A    |                              |(100.100.100.10)|
 |            |-+              리눅스 방화벽  +-----------------+
 |192.168.1.x | |               커
널 2.4.14          |          +--------------+
 +------------+ |           +---------------
+        |        |              |
                |           |(100.100.100.2)|        |        |               |
 +------------+ | +------+ |           eth0|-----------------
|              |
 |   Net-B    | | | 내
부 | |eth1            | (100.100.100.1) |              |
 |            |-+-|      |-|(192.168.1.254)|    E1 전용선    |     인터
넷    |
 |192.168.2.x | | |라우
터| |                |                  |              |
 +------------+ | +------+ |           ppp0|-----------------
|              |
                |           |(200.200.200.2)|      ADSL      |               |
 +------------+ |           +---------------
+ (200.200.200.1) |                |
 |   Net-C    | |                                           +------------
--+
 |            |-+
 |192.168.3.x |
 +------------+
```

위의 그림과 같이 내부 네트웍은 모두 세 개의 서브넷으로 나누어져 있고, 각 서브넷 간의 네트워킹은 내부 라우터를 통해서 이루어지며 인터넷을 사용하기 위해서는 내부 라우터를 거친 후 리눅스 방화벽을 통해야 한다. 즉, 리눅스 방화벽에서 IP Masquerading도 지원해야 한다. 그리고 인터넷으로 연결되는 라인은 E1 전용선과 ADSL이 있음을 알 수 있다.

각 서브넷의 default gateway는 192.168.?.1 이며, netmask는 255.255.255.0이라고 가정한다. 커널 2.4.14의 리눅스 방화벽에서 네트웍 트래픽 분산 및 패킷 필터링을 할 것이다.

4. 커널 컴파일 옵션

커널 2.4.14를 기준으로 커널 컴파일 옵션에 대해 알아본다. 여기에서 언급하는 옵션은 네트웍 트래픽 분산 및 IP Masquerading에 필요한 부분들이다. 괄호 부분에 아무런 표시가 없이 비어 있는 옵션은 반드시 필요한 것은 아니며 해도 좋고 안해도 그만인 옵션이다.

```
General setup  --->
    [*] Networking support

Networking options  --->
    <*> Packet socket
    [*]   Packet socket: mmapped IO
    [*] Kernel/User netlink socket
    [*]   Routing messages
    [*] Network packet filtering (replaces ipchains)
    [ ]   Network packet filtering debugging
    [*] Socket Filtering
    <*> Unix domain sockets
    [*] TCP/IP networking
    [*]   IP: advanced router
    [*]     IP: policy routing
    [*]       IP: use netfilter MARK value as routing key
    [*]       IP: fast network address translation
    [ ]     IP: equal cost multipath
    [*]     IP: use TOS value as routing key
    [*]     IP: verbose route monitoring
    [*]     IP: large routing tables

      IP: Netfilter Configuration  --->
          <M> Connection tracking (required for masq/NAT)
          <M>   FTP protocol support
          <M>   IRC protocol support
          <M> Userspace queueing via NETLINK (EXPERIMENTAL)
          <M> IP tables support (required for filtering/masq/NAT)
          <M>   limit match support
          <M>   MAC address match support
          <M>   netfilter MARK match support
          <M>   Multiple port match support
          <M>   TOS match support
          <M>   LENGTH match support
          <M>   TTL match support
          <M>   tcpmss match support
          <M>   Connection state match support
          <M>   Unclean match support (EXPERIMENTAL)
          <M>   Owner match support
          <M>   Packet filtering
          <M>     REJECT target support
          <M>   Full NAT
          <M>     MASQUERADE target support
```

```
<M>   Packet mangling
<M>     TOS target support
<M>     MARK target support
<M>   LOG target support
<M>   TCPMSS target support
```

Ethernet card, PPP 등에 대한 커널 옵션 설정도 잊지 말아야 할 것이다.

5. 네트웍 트래픽 분산의 실제

5.1. 설치해야 할 소프트웨어

네트웍 트래픽 분산 및 IP Masquerading, 그리고 PPP와 관련하여 리눅스 방화벽에 설치해야 할 소프트웨어를 살펴보자. 필자의 리눅스 방화벽에 설치되어 있는 소프트웨어 목록은 아래와 같다.

1. iproute-20001007
2. iptables-1.2.4
3. ppp-2.4.1
4. pppoe-3.2

iproute는 iproute2 라는 이름의 소프트웨어이며 데비안에서는 iproute 라는 이름의 패키지로 배포하고 있다. 이 소프트웨어는 기존의 route 명령을 대체하는 명령어(ip)를 포함하고 있으며 네트웍 대역폭을 조절하는 명령어(tc)도 포함되어 있다.

iptables는 커널 2.2에서 제공하던 ipchains를 대체하는 명령어로 ipchains보다 많은 기능을 제공한다.

5.2. 트래픽 분산의 순서

본론에 들어가기 전에 네트웍 트래픽을 분산시키기 위해서 밟아야 할 순서를 짚어보자.

먼저 트래픽 분산 정책을 결정해야 할 것이다. 즉, a 종류의 패킷은 1번 문으로 내보내고, b 종류의 패킷은 2번 문으로 내보낸다는 식의 정책을 결정해야 할 것이다. 이러한 정책을 결정하기 위해서는 고려해야 할 것이 몇가지 있을 것이다. 예를 들면, 끊김이 없어야 하는 패킷은 전용선 쪽으로 내보내고, 그렇지 않은 패킷은 ADSL 쪽으로 내보낸다는 식이다. 그리고 전체적인 네트웍 트래픽의 양도 감안해야 할 것이다.

그 다음, 결정된 정책에 따라서 ip 명령으로 라우팅 룰을 설정하고, 라우팅 테이블을 만들어야 한다. 그리고 내보낼 패킷과 내보내지 않을 패킷, 들어와서는 안되는 패킷 등을 구분하여 필터링을 할 예정이라면 iptables 명령으로 패킷을 어떻게 걸러낼 것인지 설정한다. 그리고 iptables 명령으로 Masquerading 설정을 한다. 아래에 그 순서를 간단하게 정리하였다.

1. 네트웍 트래픽 분산 정책을 결정한다.
2. 분산 정책에 따라서 라우팅 룰을 설정한다.
3. 분산 정책에 해당하는 라우팅 테이블을 설정한다.
4. iptables 명령으로 필터링 룰을 설정한다.
5. iptables 명령으로 Masquerading을 설정한다.

5.3. 호스트 또는 네트웍 기반의 트래픽 분산

3절에서 설명한 네트웍 환경을 기준으로 하여, 네트웍 트래픽을 서브네트웍 별로 분산시키는 상황을 가정한다. 아래는 리눅스 방화벽에 설정되어 있는 라우팅 테이블을 route 명령과 ip 명령으로 본 결과이다. 그리고 그 뒤에 라우팅 룰과 main 라우팅 테이블도 표시 하였다.

```
# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Win-
dow  irtt Iface
200.200.200.1   0.0.0.0         255.255.255.255 UH       40 0          0 ppp0
100.100.100.0   0.0.0.0         255.255.255.    U        40 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U        40 0          0 eth1
0.0.0.0         100.100.100.1  0.0.0.0         UG       40 0          0 eth0

# ip route ls
200.200.200.1  dev ppp0  proto kernel  scope link  src 200.200.200.2
100.100.100.0/24 dev eth0  proto kernel  scope link  src 100.100.100.2
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.254
default via 100.100.100.1 dev eth0

# ip rule ls
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default

# ip route ls table main
200.200.200.1 dev ppp0  proto kernel  scope link  src 200.200.200.2
100.100.100.0/24 dev eth0  proto kernel  scope link  src 100.100.100.2
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.254
default via 100.100.100.1 dev eth0
```

ip rule ls 명령의 결과를 보면 기본적으로 세개의 라우팅 테이블 local, main 그리고 default가 있음을 알 수 있다. 각 라우팅 테이블의 자세한 내용은 **ip route ls table local**과 같은 명령으로 알 수 있다. 여러분 각자 확인해 보기 바란다. 위에서 보다시피 각 라우팅 테이블 에는 번호가 붙어 있으며 리눅스 커널은 낮은 번호 순서대로 패킷을 처리한다. 즉 패킷이 네트웍을 통해서 들어오면 커널은 그 패킷이 어디로 향하는지 확인하고서는 어떻게 라우팅할 지 local 테이블

에서 먼저 찾아본다. local 테이블에 적당한 처리 규정이 없으면 main 테이블을 찾아보고 거기에도 없으면 마지막으로 default 테이블을 찾아보는 것이다.

from all이 의미하는 바는 패킷이 어디에서 온 것인지를 정의하는 것이다. all 이므로 어디에서 온 것이든 상관없다는 의미이다. **from all**을 **from 192.168.1.100**으로 바꾸면 **192.168.1.100**에서 출발한 패킷을 의미하게 된다. 즉 출발지에 따라서 라우팅을 결정할 수 있는 것이다. 이를 두고 **Source Routing**이라 한다.

ip route ls table main 명령의 실행 결과를 살펴보면, main 테이블에 ppp0에 대한 라우팅이 설정되어 있고, **E1** 전용선에 대한 라우팅이 **default**로서 설정되어 있음을 알 수 있다. **netstat -rn** 명령의 결과와 같음을 알 수 있다. 즉, **netstat -rn** 명령은 main 테이블을 표시한 것이다.

이제 5.2절트래픽 분산의 순서에서 설명한 대로 하나씩 해보자.

네트웍 트래픽의 분산 정책 결정

1. Net-A와 Net-B로부터 인터넷으로 향하는 패킷은 E1 전용선을 통하여 나간다.
2.

   Net-C로부터 인터넷으로 향하는 패킷은 ADSL(ppp0)을 통하여 나간다.

라우팅 룰의 설정

1. `/etc/iproute2/rt_tables`에 적당한 라우팅 테이블 이름을 넣어 준다. 라우팅 테이블 이름을 e1line, adslline 이라고 하자.

   ```
   # echo 201 e1line >> /etc/iproute2/rt_tables
   # echo 202 adslline >> /etc/iproute2/rt_tables
   ```

2. 분산 정책에 따라서 라우팅 룰을 추가한다.

   ```
   # ip rule add from 192.168.1.0/24 table e1line
   # ip rule add from 192.168.2.0/24 table e1line
   # ip rule add from 192.168.3.0/24 table adslline
   ```

라우팅 테이블의 설정

e1line과 adslline 라우팅 테이블을 만든다.

```
# ip route add default via 100.100.100.1 dev eth0 table e1line
# ip route add default via 200.200.200.1 dev ppp0 table adslline
```

여기서 잠깐

필터링 룰을 설정하기 전에 설정된 라우팅 룰을 살펴보고 고려해야 할 사항을 알아보자.

```
# ip rule ls
0:      from all lookup local
```

```
32763:  from 192.168.3.0/24 lookup adslline
32764:  from 192.168.2.0/24 lookup e1line
32765:  from 192.168.1.0/24 lookup e1line
32766:  from all lookup main
32767:  from all lookup default
```

라우팅 룰의 번호 및 순서를 살펴보면 위에서 실행시킨 **ip rule add** 명령의 순서와 역순임을 알 수 있을 것이다. 염두에 두기 바란다. 이제 예를 들면서 고려해야 할 사항을 알아보자.

192.168.1.10에서 **111.111.111.100** 으로 향하는 패킷을 생각해 보자. 그 패킷이 리눅스 방화벽에 들어가면 리눅스 커널은 0번 라우팅 룰부터 대조하여 어떻게 라우팅 할 것인지 결정할 것이다. 0번 라우팅 룰은 그 패킷이 어디에서 출발 한 것인지 상관없이 (from all) local 라우팅 테이블을 참조하도록 되어 있다. 그러나 local 테이블을 살펴보면 알겠지만 거기에는 **111.111.111.100**으로 향하는 패킷에 대하여 마땅히 처리할 라우팅 규칙이 없다. 그 다음 라우팅 룰(32763번)은 **192.168.3.0**네트웍으로부터 출발하는 패킷에 대한 것이므로 해당 사항이 없다. 32764번 룰도 비슷하다. 32765번 룰은 **192.168.1.0** 네트웍으로부터 출발하는 패킷에 대하여 적용 가능하다. e1line 라우팅 테이블을 살펴보자.

```
# ip route ls table e1line
default via 100.100.100.1 dev eth0
```

패킷이 어디로 향하든지 상관없이(default) **100.100.100.1** 을 거쳐서 패킷을 라우팅 하도록 설정되어 있음을 알 수 있다. 따라서 **192.168.1.10**에서 **111.111.111.100** 으로 향하는 패킷은 32765번 룰을 적용할 수 있는 것이다. 그런데 이 절의 앞부분을 다시 읽어보면 알겠지만, e1line 테이블에 있는 라우팅 규칙과 똑같은 것이 main 테이블에도 있다. 다시 생각해 보면 **192.168.1.0**네트웍으로부터 출발하는 패킷의 라우팅은 main 테이블에서 처리할 수 있다는 것이다. 즉, e1line 테이블을 굳이 추가 할 필요가 없다는 것이다.

또 다른 경우를 살펴보자. **192.168.3.10**에서 웹서버인 **100.100.100.10** 으로 향하는 패킷을 생각해 보자. 앞에서 살펴본 과정과 비슷하게 따라가 보면 이 패킷은 ADSL을 통해서 빠져 나간 후 인터넷을 거쳐서 웹서버로 들어옴을 알 수 있다. eth0를 통해서 빠져 나가면 바로 갈 수 있는 길을 크게 돌아서 온 것이다. 따라서 이에 대한 보완도 필요함을 알 수 있다. ADSL도 비슷한 상황을 가정할 수 있으나 여기서는 굳이 고려하지 않겠다. 왜냐하면 귀찮으니까.... 그리고 생각해 본 적도 없으므로.... 게다가 별 필요가 없으므로...

한 가지 더 고려해야 할 사항은 리눅스 방화벽에서 **192.168.1.0** 네트웍으로 나가는 라우팅 규칙은 local 테이블에 명시되어 있지만 **192.168.2.0**, **192.168.3.0** 네트웍으로 나가는 라우팅 규칙은 명시되어 있지 않다. 이를 local 테이블에 추가해 주어야 한다. 왜 local 테이블에 추가하는 지는 생각해 보면 알 수 있을 것이다.

위의 상황을 고려하여 라우팅 룰과 라우팅 테이블을 다시 만들어 보자. 앞에서 `/etc/iproute2/rt_tables`에 **echo** 문으로 추가했던 두 줄을 지운 후 아래 명령을 실행해야 할 것이다.

```
# echo 201 adslline >> /etc/iproute2/rt_tables
# ip rule add from 192.168.3.0/24 table adslline
# ip route add default via 200.200.200.1 dev ppp0 table adslline
# ip route add 100.100.100.0/24 via 100.100.100.2 dev eth0 table local
# ip route add 192.168.2.0/24 via 192.168.1.254 dev eth1 table local
# ip route add 192.168.3.0/24 via 192.168.1.254 dev eth1 table local
```

마지막 두 줄은 **192.168.0.0** 네트웍에 대한 라우팅 규칙으로 생각해서 한 줄로 표현할 수 있다.

```
# echo 201 adslline >> /etc/iproute2/rt_tables
# ip rule add from 192.168.3.0/24 table adslline
# ip route add default via 200.200.200.1 dev ppp0 table adslline
# ip route add 100.100.100.0/24 via 100.100.100.2 dev eth0 table local
# ip route add 192.168.0.0/16 via 192.168.1.254 dev eth1 table local
```

필터링 룰의 설정

필터링 룰을 설정하는 것은 여기서 생략한다.

Masquerading의 설정

전용선 쪽으로 나가는 패킷과 ADSL 쪽으로 나가는 패킷에 대한 Masquerading 설정이 필요할 것이다.

```
# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j SNAT --to 100.100.100.2
# iptables -t nat -A POSTROUTING -o ppp0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j MASQUERADE
```

이것으로 호스트 또는 서브네트웍 별로 네트웍 트래픽을 분산시키는 방법을 알아 보았다. 아래에 명령어를 다시 한 번 정리하였다.

```
# echo 201 adslline >> /etc/iproute2/rt_tables
# ip rule add from 192.168.3.0/24 table adslline
# ip route add default via 200.200.200.1 dev ppp0 table adslline
# ip route add 100.100.100.0/24 via 100.100.100.2 dev eth0 table local
# ip route add 192.168.0.0/16 via 192.168.1.254 dev eth1 table local
# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j SNAT --to 100.100.100.2
# iptables -t nat -A POSTROUTING -o ppp0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j MASQUERADE
```

5.4. port 기반의 트래픽 분산

port를 구분해서 패킷을 내보내는 방법과 호스트 또는 서브네트웍을 구분해서 패킷을 내보내는 방법은 한 가지를 제외하고는 큰 차이가 없다. 기본적으로 커널은 어떤 패킷이 어디로 향하는 지를 보고서 라우팅을 어떻게 할 지를 결정하며, 그 패킷이 www port를 사용하는지, telnet port를 사용하는지 등은 라우팅의 기준이 되지 않는다. 따라서 port를 구분해서 라우팅을 하기 위해서는 또 다른 기준을 만들어 주어야 하며, 그 방법은 특정 port를 사용하는 패킷에 표식을 해서 구분하는 것이다. 패킷에 표식을 하는 것은 **iptables** 명령을 이용한다.

네트웍 트래픽의 분산 정책 결정

1. www, ftp, ftp-data port를 사용하는 패킷은 ADSL(ppp0)을 통하여 나간다. 나간다.
2. 이외의 나머지 패킷은 E1 전용선을 통하여 나간다.

라우팅 룰의 설정

1. `/etc/iproute2/rt_tables`에 ADSL을 통해서 나가는 패킷을 위해 adslline 라우팅 테이블 이름을 추가한다.

   ```
   # echo 201 adslline >> /etc/iproute2/rt_tables
   ```

2. 분산 정책에 따라서 라우팅 룰을 추가한다.

   ```
   # ip rule add fwmark 1 table adslline
   ```

위 명령의 의미는 "1"이라고 표시된 패킷은 adslline 테이블을 참조하여 라우팅한다라는 뜻이다. 트래픽 분산 정책에 따라서 www, ftp, ftp-data port를 사용하는 패킷에 대하여 다음과 같이 "1"이라고 표시하면 되는 것이다.

```
# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport www -j MARK --set-mark 1

# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport ftp -j MARK --set-mark 1

# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport ftp-data -j MARK --set-mark 1
```

첫번째 명령을 해석해 보면, eth1를 통해서 들어오는 패킷 중에서 출발지가 192.168.0.0/16 네트웍이고 목적지는 192.168.0.0/16이 아니며 www port를 사용하는 패킷에 대하여 "1"이라고 표시한다라는 의미이다.

라우팅 테이블의 설정

adslline 라우팅 테이블을 만들고 local 테이블에 필요한 라우팅 규칙을 추가한다.

```
# ip route add default via 200.200.200.1 dev ppp0 table adslline
# ip route add 100.100.100.0/24 via 100.100.100.2 dev eth0 table local
# ip route add 192.168.0.0/16 via 192.168.1.254 dev eth1 table local
```

필터링 룰의 설정

필터링 룰을 설정하는 것은 여기서 생략한다.

Masquerading의 설정

5.3절에서와 마찬가지로 전용선 쪽으로 나가는 패킷과 ADSL 쪽으로 나가는 패킷에 대한 Masquerading 설정이 필요할 것이다.

```
# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j SNAT --to 100.100.100.2
# iptables -t nat -A POSTROUTING -o ppp0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j MASQUERADE
```

명령어를 다시 한 번 정리하면,

```
# echo 201 adslline >> /etc/iproute2/rt_tables
# ip rule add fwmark 1 table adslline
# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport www -j MARK --set-mark 1
# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport ftp -j MARK --set-mark 1
# iptables -A PREROUTING -i eth1 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 \
-t mangle -p tcp --dport ftp-data -j MARK --set-mark 1
# ip route add default via 200.200.200.1 dev ppp0 table adslline
# ip route add 100.100.100.0/24 via 100.100.100.2 dev eth0 table local
# ip route add 192.168.0.0/16 via 192.168.1.254 dev eth1 table local
# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j SNAT --to 100.100.100.2
# iptables -t nat -A POSTROUTING -o ppp0 -s 192.168.0.0/16 ! -
d 192.168.0.0/16 -j MASQUERADE
```

한가지 더

위와 같이 port 별로 네트웍 트래픽을 분산시키려 할 때 전혀 라우팅이 안되는 경우가 있다. 아마도 대부분의 리눅스 배포본에서 이런 현상이 발생할 것으로 생각된다. 이는 spoof protection을 위해 `/proc/sys/net/ipv4/conf/*/rp_filter` 파일의 값을 "1"로 설정했기 때문이다. 모든 디바이스에 대하여 `rp_filter` 파일의 값을 "0"으로 설정하면 문제가 해결될 것이다.

6. 백업 라인에 대하여

이 문서에서 다루고 있는 전용선 + ADSL과 같은 환경의 경우, ADSL의 접속이 끊길 때가 있다. 현재와 같은 네트웍 환경이라면 ADSL 접속이 끊기더라도 전용선을 통해서 인터넷을 사용할 수는 있다. adslline 라우팅 테이블에서 라우팅 규칙을 찾지 못하면 main 라우팅 테이블에서 라우팅 규칙을 찾기 때문이다. 그러나, ADSL은 살아 있고 전용선이 끊어지는 경우에는 전용선을 타고 나가야 할 패킷을 어떻게 라우팅 할 것인지를 알 수가 없게 된다. 패킷을 어떻게 라우팅할 지는 adslline 라우팅 테이블을 검색한 다음 적당한 라우팅 규칙이 없으면 main 라우

팅 테이블에서 규칙을 검색할텐데, main 테이블에는 ADSL을 이용하는 적당한 라우팅 규칙이 없기 때문이다. 이 같은 경우에, main 라우팅 테이블에 ADSL과 관련된 라우팅 규칙을 넣어 놓았다면 괜찮을 것이다. 즉, 백업 라인을 구성하려면 adslline 과 main 라우팅 테이블에 서로에 대한 라우팅 규칙을 추가하는 것이다. **ip route** 명령을 사용할 때 metric 값을 달리하여 서로에 대한 라우팅 규칙을 넣어주면 된다.

adslline 라우팅 테이블에는,

```
# ip route add default via 200.200.200.1 dev ppp0 table adslline metric 1
# ip route add default via 100.100.100.1 dev eth0 table adslline metric 2
```

main 라우팅 테이블에는,

```
# ip route add default via 200.200.200.1 dev ppp0 table main metric 2
```

직접 시험해 보기 바란다.

7. 스크립트 예

5.3절에서 설명한 내용을 기준으로 스크립트를 작성하였으며, 이를 적용하기 위해서는 여러분 각자의 환경에 맞게 수정할 필요가 있을 것이다. 아래의 스크립트는 라우팅 테이블을 생성하는 역할을 하며 IP Masquerading 역할은 하지 않는다는 것을 주의하기 바란다. 따라서 IP Masquerading을 구현하는 스크립트를"Linux IP Masquerade HOWTO" 문서를 참고로하여 따로 작성하기 바란다. 아래의 스크립트도 "Linux IP Masquerade HOWTO" 문서를 참고로하여 작성하였다.

```
#! /bin/sh

KERNELVER="`uname -r | cut -d. -f1,2`"

if [ "$KERNELVER" = "2.2" ]
then
    echo "This does not support the Kernel 2.2.x"
    exit 1
fi

IPTABLES=/sbin/iptables
IFCONFIG=/sbin/ifconfig
ECHO=/bin/echo
GREP=/bin/grep
AWK=/usr/bin/awk
SED=/bin/sed
IP=/sbin/ip
RTTAB=/etc/iproute2/rt_tables

ONLINEPPP0=`$IFCONFIG | $GREP ppp0 | $AWK '{print $1}'`
```

```
if [ -n "$ONLINEPPP0" ]
then
     $ECHO "PPP0 is ON-LINE...   Initializing routing table......"
else
     $ECHO "PPP0 is OFF-LINE..."
     exit 1
fi

# 내부 네트웍에 연결된 이더넷 카드
INTIF="eth1"

# 인터넷에 연결된 이더넷 카드
EXITIFETH="eth0"

#ppp 연결
EXITIFPPP0="ppp0"
ALLPPP="ppp+"

# eth1에 할당된 ip address
INTIP="`$IFCONFIG $INTIF | $GREP 'inet addr' | $AWK '{print $2}' | $SED -
e 's/.*://'`"

# eth0에 할당된 ip address
FWIP="`$IFCONFIG $EXI-
TIFETH | $GREP 'inet addr' | $AWK '{print $2}' | $SED -e 's/.*://'`"

# ppp0에 할당된 ip address
PPP0IP="`$IFCONFIG $EXITIF-
PPP0 | $GREP 'inet addr' | $AWK '{print $2}' | $SED -e 's/.*://'`"

# E1 전용선 게이트웨이
FWGW="100.100.100.1"

# ppp0 게이트웨이
PPP0GW="`$IFCONFIG $EXITIF-
PPP0 | $GREP 'inet addr' | $AWK '{print $3}' | $SED -e 's/.*://'`"

# 내부 네트웍 및 전용선 네트웍
NETA="192.168.1.0/24"
NETB="192.168.2.0/24"
NETC="192.168.3.0/24"
NETALL="192.168.0.0/16"
NETE1="100.100.100.0/24"
UNIVERSE="0.0.0.0/0"

add_rt_table() {
    # rt_tables 파일 생성
    $ECHO 255     local > $RTTAB
    $ECHO 254     main >> $RTTAB
```

```
    $ECHO 253       default >> $RTTAB
    $ECHO 0         unspec >> $RTTAB
    $ECHO 1         inr.ruhep >> $RTTAB

    # 사용할 라우팅 테이블 이름 삽입
    $ECHO 201 adslline >> $RTTAB

    # 라우팅 룰 추가
    $IP rule add from $NETC table adslline

    # 라우팅 테이블 생성
    $IP route add default via $PPP0GW dev $EXITIFPPP0 table adslline

    # 내부 네트웍과 E1 전용선 네트웍에 대한 라우팅을 local table에 추가
    $IP route add $NETE1 via $FWIP dev eth0 table local
    $IP route add $NETALL via $INTIP dev eth1 table local
}

del_rt_table() {
    # 라우팅 테이블과 룰 삭제
    $IP route del $NETALL via $INTIP dev $INTIF table local
    $IP route del $NETE1 via $FWIP dev $EXITIFETH table local
    $IP route del default via $PPP0GW dev $EXITIFPPP0 table adslline
    $IP rule del from $NETC table adslline
}

case "$1" in
    start)
        add_rt_table
        ;;
    stop)
        del_rt_table
        ;;
    restart)
        del_rt_table
        add_rt_table
        ;;
    *)
        echo "Usage: 스크립트 파일 이름 {start|stop|restart}"
        exit 1
        ;;
esac

exit 0
```

8. 덧붙이는 말

이왕이면 패킷 필터링 등에 대한 언급도 하고 싶었으나 너무 내용이 많아지고 다루고자 했던 내용의 범위를 벗어나므로 여기서는 언급하지 않았다. 하지만 다른 여러 문서에서 내용을 다루고 있으므로 참고하기 바란다.

ADSL 사용자를 위하여 마지막으로 한 가지 덧붙이겠다. ADSL을 사용하여 Masquerading을 할 때 특정 사이트에 접속이 안되는 문제가 있을 수 있다. 이 때는 **ppp**에서 **pppoe**를 실행시킬 때 **-m 1412** 옵션을 주면 아마도 잘 될 것이다.

네트웍 트래픽 분산을 위해서 설명한 내용 그대로 시험한 것은 아니며 특히 port에 따라서 트래픽을 분산하는 것은 정확히 테스트된 것이 아니다. 틀린 내용이 있을 수 있으니 미리 알아두기 바란다.

9. 참고자료

아래의 참고 자료 대부분은 리눅스 한글 문서 프로젝트 홈페이지 (http://kldp.org)에서 한글 번역자료를 찾을 수 있을 것이다.

1. Linux Advanced Routing and Traffic Control HOWTO (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Adv-Routing-HOWTO.html)
2. Linux IP Masquerade HOWTO (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/IP-Masquerade-HOWTO.html)
3. Linux 2.4 NAT HOWTO (http://netfilter.samba.org/documentation/HOWTO/NAT-HOWTO.html)
4. Linux 2.4 Packet Filtering HOWTO (http://netfilter.samba.org/documentation/HOWTO/packet-filtering-HOWTO.html)