네임스페이스 (Namespaces) 운영, part 7: Network namespaces

By Jake Edge
January 22, 2014
번역 by Ian Y. Choi, NAIM Networks on
March 27, 2014

Namespaces in operation

기사 원문을 보시려면 클릭하세요.
(Please click this URL if you want to read the original article written in English. Thanks.)

lwn.net 에서 Linux namespace 를 다룬지 꽤 되었다. 본 시리즈에서 빠졌던 '네트워크 네임스페이스' 부분을 이제야 채우고자 한다. 이름에서 알 수 있듯이, 네트워크 네임스페이스는 네트워크 장치, 주소, 포트, 라우트, 방화벽 규칙 등의 사용을 각각 분할하여, 별도의 상자(박스)처럼 분리한다. 이를 통해 단일 커널 인스턴스가 실행 중인 환경에서 네트워크를 가상화하는 것이 가능해진다. 네트워크 네임스페이스는 이미 5 년 가까이 지난 커널 버전 2.6.24 에 추가되었다. (현재와 같이) 자주 쓰일 정도로 준비된 상황까지 발전하기까지는 1 년 정도 소요되었는데, 이 때 이후, 네트워크 네임스페이스는 많은 개발자로부터 많이 간과된 측면이 있다.

네트워크 네임스페이스 관리 기본

다른 네임스페이스들과 비슷하게, 네트워크 네임스페이스 역시 CLONE_NEWNET 플래그값을 clone() 시스템 콜에 전달하는 과정을 통해 생성된다. 그러나, 명령 라인 방식으로 실행 가능한 ip 라는 네트워크 구성 도구를 사용하여 네트워크 네임스페이스를 셋업하고 작업하는 것 또한 가능하다. 예를 들면,

```
# ip netns add netns1
```

위 명령어는 netns1 라는 새로운 네트워크 네임스페이스 1 개를 생성한다. ip 도구가 네트워크 네임스페이스를 하나 생성할 때, /var/run/netns 아래에 해당 네임스페이스를 위한 연결 마운트 지점이 생성될 것이다. 이를 통해, 프로세스들이 해당 네임스페이스 안에 없더라도 네임스페이스가 지속될 수 있도록 하고, 네임스페이스 자체를 변경하는 것을 가능하게 한다. 네트워크 네임스페이스의 경우 사용 전 어느 정도 분량이 되는 구성을 일반적으로 필요로 하기에, 이 특징은 시스템 관리자들에게 많은 도움을 줄 것이다.

"ip netns exec" 명령은 네임스페이스 내에서 네트워크 관리 명령어들을 실행하는데 사용된다.

```
# ip netns exec netns1 ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

이 명령어는 네임스페이스 내에서 보이는 인터페이스들을 열거한다.
네임스페이스는 다음을 통해 제거 가능하다.

```
# ip netns delete netns1
```

이 명령어는 주어진 네트워크 네임스페이스와 연관된 연결 바운트 지점을 제거한다. 그러나, 네임스페이스 자체는 해당 네임스페이스 내 실행되는 프로세스가 실행될 때까지 계속 지속될 것이다.

네트워크 네임스페이스 구성

새 네트워크 네임스페이스에는 다른 네트워크 장치들은 존재하지 않고 루프백 장치(loopback device) 1 개만 있을 것이다. 루프백 장치를 제외하고, 각 네트워크 장치(물리 또는 가상 인터페이스, 브릿지 등)는 오직 1 개의 단일 네트워크 네임스페이스에만 소속된다. 게다가, (실제 하드웨어에 연결된) 물리 장치들은 root 네임스페이스가 아닌 어떤 네임스페이스에도 할당되어질 수 없다. 대신, 가상 네트워크 장치들은 (예: 가상 ethernet, 즉 veth) 생성되어 네임스페이스에 소속될 수 있다. 이 가상 장치들은 프로세스들이 네임스페이스 내에서만 네트워크 통신을 수행하도록 지원한다. 이를 통해 누구와 통신을 할 수 있는지에 대한 대상을 결정하는 구성, 라우팅 등이 이루어진다고 볼 수 있다.

처음 생성되었을 때, 새 네임스페이스 상에 있는 lo 루프백 장치는 down 상태이므로, 루프백 장치에 ping 을 수행하면 실패할 것이다.

```
# ip netns exec netns1 ping 127.0.0.1
connect: Network is unreachable
```

해당 인터페이스를 up 을 시키는 과정을 통해 루프백 주소에 ping 하는 것이 가능해진다.

```
# ip netns exec netns1 ip link set dev lo up
# ip netns exec netns1 ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.051 ms
...
```

그러나 여전히 netns1 과 root 네임스페이스와는 통신이 불가능하다. 이를 지원하기 위해서는, 가상 ethernet 장치를 생성 및 구성할 필요가 있다.

```
# ip link add veth0 type veth peer name veth1
```

```
# ip link set veth1 netns netns1
```

처음 명령어는 가상 ethernet 장치 한 쌍을 연결된 상태로 만든다. veth0 로 보내지는 패킷은 veth1 에서 받을 것이고, 그 반대 또한 동작할 것이다. 두 번째 명령어는 veth1 을 netns1 네임스페이스에 할당한다.

```
# ip netns exec netns1 ifconfig veth1 10.1.1.1/24 up
# ifconfig veth0 10.1.1.2/24 up
```

그리고 나서, 이 두 명령어를 통해 IP 주소를 두 장치에 할당한다.

```
# ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.087 ms
...

# ip netns exec netns1 ping 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=0.054 ms
...
```

위에서 보여준 ping 명령어에서 보듯이, 양방향 통신이 이제 가능해진다.

언급하였듯이, 그러나, 네임스페이스들은 라우팅 테이블 또는 방화벽 규칙을 서로 공유하지 못한다. netns1 에서 route 와 iptables -L 명령을 실행함으로써 증명될 것이다.

```
# ip netns exec netns1 route
# ip netns exec netns1 iptables -L
```

첫 번째 명령은 단순히 (veth1 이 사용하는) 10.1.1 서브넷을 위한 패킷들의 경로를 보여줄 것이고, 반명 두 번째 명령은 iptables 구성이 되어 있지 않음을 보여줄 것이다. 두 결과 모두 의미하는 바는, netns1 으로부터 인터넷으로 보내지는 다량의 패킷들은 "Network is unreachable"라는 두려운 메시지를 수신할 것이다. 필요한 경우, 네임스페이스를 인터넷에 연결하는 몇 가지 방법이 있다. bridge 를 root 네임스페이스와 netns1 에 있는 veth 장치에 생성할 수 있을 것이다. 다른 방법으로는, 네트워크 주소 변환 (NAT) 과 결합된 IP 포워딩을 root 네임스페이스에 구성할 수 있다. 이들 중 어떤 방식이든 (그리고 다른 구성할 수 있는 여러 방법들이 있을 것이다.) 패킷들이 netns1 으로부터 인터넷에 도달하도록 하고 해당 패킷들에 대한 응답이 netns1 에 도달하도록 하는 것이 가능할 것이다.

네임스페이스에 할당된 (clone(), unshare(), 또는 setns()를 통해) root 권한으로 실행되지 않는 프로세스들은 이미 셋업된 네트워크 장치 및 구성에 대해서만 접근 가능하다-물론, root 사용자는 새로운 장치들을 추가하고 구성할 수 있다. ip netns 서브 명령을 사용하여, 네트워크 네임스페이스를 지칭(address)하는 두 가지 방법이 존재한다. 그 중 하나는 netns1 과 같은 이름을 사용하는 것이고, 다른 하나는 해당 네임스페이스 내에 있는 프로세스의 프로세스 ID (PID)을 통한 방법이다. init 이 일반적으로 root 네임스페이스 내에 존재하므로, 다음과 같은 명령어를 사용할 수 있다.

```
# ip link set vethX netns 1
```

저 명령어를 통해 (짐작컨데, 새로 생성된) veth 장치를 root 네임스페이스에 위치시키고, 다른 네임스페이스 상에서 root 사용자가 동작시킬 수 있을 것이다. root 사용자가 네트워크 네임스페이스 내에서 이와 같은 운영을 수행 가능하게하는 것이 적절하지 않은 상황일 수도 있겠지만, PID 와 네임스페이스 마운트 기능은 다른 네트워크 네임스페이스들에 도달하지 못하도록 사용될 수도 있을 것이다.

네트워크 네임스페이스 사용

이제까지 살펴보았듯이, 네임스페이스의 네트워크 동작은 아무것도 동작하지 않는 상태 (즉, 단순히 루프백만 존재하는 경우)부터 시스템 네트워크 동작에 대한 모든 접근까지 가능하도록 할 수 있다. 이를 통해 수많은 다양한 네트워크 네임스페이스를 사용한 유스케이스가 있을 것이다.

본질적으로 네임스페이스 내에서 네트워크를 끔으로써, 관리자들은 해당 네임스페이스 내에서 실행되는 프로세스들이 네임스페이스 밖으로 접속을 생성하지 못하도록 보장할 수 있다. 해당 프로세스가 보안 취약점과 같은 과정을 통해 감염되더라도, 네임스페이스는 botnet 에 참여하거나 스팸을 보내는 것과 같은 행동들을 못하도록 해줄 것이다.

심지어 네트워크 트래픽을 다루는 프로세스들 (예: 웹 서버 worker 프로세스 또는 웹 브라우저 렌더링 프로세스) 또한 제한된 네임스페이스로 위치시킬 수 있다. 외부 지점에 의해, 또는 외부 지점으로 향하는 접속 하나가 생성되면, 해당 접속을 위한 파일 디스크립터 (file descriptor)는 clone() 시스템 콜에 의해 생성된 새로운 네트워크 네임스페이스의 자식 프로세스에 의해 관리 가능하다. 자식은 부모의 파일 디스크립터들을 모두 상속받을 것이기에, 해당 자식 프로세스는 접속된 디스크립터에 대한 접근 권한이 있다. 또 다른 가능성으로는 부모가 파일 디스크립터들을 제한된 네트워크 네임스페이스 내에 있는 프로세스에 Unix 소켓을 통해 보낼 수도 있다. 어떤 경우라도, 적절한 네트워크 장치가 해당 네임스페이스에 존재하지 않으면 자식 또는 worker 프로세스가 부가적인 네트워크 접속을 생성하는 것이 불가능할 것이다.

네임스페이스는 모든 것들이 단일 박스에서 동작하는 다소 복잡한 네트워킹 구성을 필요로 하는 환경에서 테스트하는 데 사용할 수도 있다. 보다 락-다운인 상황에서 실행되어야 하는 민감한 서비스들, 그리고 방화벽이 제한된 네임스페이스 또한 해당될 수 있다. 분명한 것은, 컨테이너 방식의 구현 또한 네트워크 네임스페이스를 사용하여 각 컨테이너에 각 네트워크만의 뷰를 제공하고, 해당 컨테이너 바깥과는 자유로운 공간을 만든다는 것이다. 이와 같이 한다면, 다양한 유스케이스들이 탄생할 수 있을 것이다.

일반적으로 네임스페이스는 시스템 자원을 분할하고, 프로세스를 그룹별로 묶어 다른 자원들로부터 격리시키는 방법을 제공한다고 할 수 있다. 네트워크 네임스페이스 역시 다른 많은 부분의 네임스페이스와 동일하지만, 네트워킹이

보안적으로 민감한 부분에 해당하기에, 여러 방법을 사용한 네트워크 격리를 제공하는 것은 굉장히 가치가 있을 것이다. 물론, 여러 네임스페이스 유형을 함께 사용한다면 보안 및 다른 필요성을 위한 보다 나은 격리를 제공할 것이다.