미래인터넷의 네트워크 가상화 기술 동향

Technical Trends of Network Virtualization in Future Internet

김영화 (Y.H. Kim) 미래네트워크연구부 책임연구원

목 차

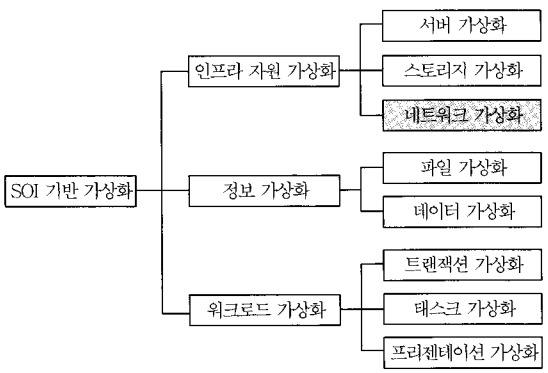
Ⅰ. 개요
Ⅱ. 네트워크 가상화의 기술 동향
Ⅲ. 결론

IT 인프라를 통해 변화하는 비즈니스 우선순위에 따라 재사용 또는 결합 가능한 컴포넌트로 통합하는 표준화된 프레임워크인 서비스 지향 아키텍처를 구현하기 위한 핵심 기술이 가상화이다. 이 가상화 기술 가운데 네트워크 가상화가 미래인터넷에서 중요한 기술적 이슈로 부상하고 있지만, 아직은 초기 단계이기 때문에 네트워크 가상화의 개념과 세부 기술 등이 모호한 상태이다. 이에 따라, 본 고에서는 미래인터넷 관점에서 네트워크 가상화에 대해 기술 동향을 파악하고, 전반적인 개념과 세부 기술을 다루어 보고자 한다.

Ⅰ. 개요

"현재 인터넷 구조의 한계성을 극복하고 미래의 새로운 요구사항을 수용하기 위해, 기존 인터넷과의 호환성을 필수 조건으로 고려하지 않는 혁신적인 개념(clean-slate)으로 설계될 미래의 새로운 인터넷"으로 시작한 미래인터넷은 대체적으로 인프라, 아키텍처, 그리고 서비스로 대별할 수 있다(나라 또는 지역마다 미래인터넷의 정의가 다를 수 있음). 현재로서는 미래인터넷의 구체적인 모습을 제시할 수 없는 상황으로 인프라, 아키텍처, 그리고 서비스 각각이 자신들의 얼굴을 다듬어 가고 있으며, 또한 이들을 서로 얽어 미래인터넷 모습을 자유롭게 그려보고 있다. 예를 들면, 미래인터넷 인프라 부분에 대한 노력으로, 테스트베드를 통해 다양한 아키텍처 및 서비스를 실험할 수 있는 환경을 구축하기 위한 R&D가 진행되고 있다. 본 고에서는 미래인터넷 인프라의 핵심 기술이 될 가능성이 높고 아직은 초기 상태에 있는 네트워크 가상화(network virtualization) 기술에 대해 미래인터넷 관점에서 기술 동향, 개념 그리고 세부 기술 등을 언급한다.

가상화 기술이란 물리적 특성을 추상화한 컴퓨팅 자원과 대상(사용자, 응용, 시스템 등) 간에 상호작용을 효율적으로 제어하는 기술이라고 말할 수 있는데, CPU, 메모리, I/O 디바이스 등과 같은 하드웨어 자원들과 단말, 애플리케이션 및 관리 등 거의 전 영역에 걸쳐 적용이 가능하다고 볼 수 있다. 이러한 가상화 기술의 분류는 보는 각도에 따라 다양하게 분류할 수 있다. 이를 테면, (그림 1)과 같이 서비스 지향 인프라 계층도에 따른 분류가 있을 수 있고, 그리고 스토리지, 서버, 네트워크 및 서비스와 같은 기술의 적용 기준에 따른 분류 등이 있을 수 있다[1],[2].



(그림 1) 서비스 지향 인프라 계층도에 따른 가상화 기술의 분류

[3]에 따르면, 가상화 기술의 적용으로 자원 활용률의 제고, 관리 비용의 감소, 사용의 유연성, 가용성 향상 등의 효과를 얻을 수 있다고 하는데, 이는 다음과 같은 폭넓은 활용이 가능하였기 때문에 발생하는 효과로 볼 수 있다.

① 동일 컴퓨터에서 다양한 서비스들의 통합
② 부하 균등
③ 효율적 전력 사용
④ 서비스 방화벽
⑤ 서로 다른 OS의 사용

이들 가운데, 다양한 서비스들의 통합, 부하 균등, 효율적 전력 사용, 서비스 방화벽은 서버에서의 가상화 활용으로 볼 수 있고, 서로 다른 OS의 사용은 데스크톱이나 노트북 등과 같은 단말에서의 가상화 활용으로 볼 수 있다. 하지만, 네트워크 가상화를 위해 추가할 수 있는 것이 바로 "다양한 응용들의 실험"이다. 특히, 미래사회의 통신 인프라를 설계해 나가는 과정에서 실제 네트워크 인프라에 준하는 대규모 시험 환경이 필요하며, 미래인터넷 테스트베드에서 IP 및 non-IP의 다양한 실험들이 서로 영향을 주지 않고 실행할 수 있게 하는 방법이 네트워크 가상화 기술이다.

네트워크 가상화는 공유하는 서브스트레이트(데이터 평면) 위에서 자신의 가상 네트워크를 동적으로 구축하여 혁신적인 네트워크 아키텍처 및 서비스 기술을 시험하고 전개하기 위한 방법을 제공한다. 이러한 네트워크 가상화 기술을 적용하게 되면 하나의 물리적인 네트워크 위에서 다양한 가상 네트워크가 존재하여 각각의 가상 네트워크가 특정 목적의 기능을 수행하면서도 가상 네트워크들간 자원은 격리된 채로 동작하여 서로에게 영향을 주지 않으면서, 필요하면 가상 네트워크들의 상호 연동도 수행한다.

이러한 방향의 첫번째 시도라고 할 수 있는 미래

인터넷 테스트베드 플랫폼은 4가지 특성을 포함할 수 있는데, 이것은 가상 네트워크간 분리, 다양한 사용자 요구사항을 수용할 수 있는 주문형 가상 네트워크의 구성, 고속 패킷 처리 능력, 그리고 저비용을 의미한다. 이들 4가지 요소는 서로 강하게 연관되어 있다. 이를 테면, 고속으로 패킷을 처리할 수 있는 성능을 보장하기 위해서는 어느 정도 규모 있는 하드웨어 소요 비용이 발생할 수 있으며, 현재 기술 수준상 가상 네트워크의 완벽한 분리 대신 부분적 분리만으로 시스템을 구성할 가능성이 높다. 하지만, 만일 미래인터넷 테스트베드를 구축하는 과정에서 기능성 및 유용성에 비해 하드웨어 및 소프트웨어의 소용 비용이 과다할 경우에는 해당 플랫폼의 광역적인 도입이 어려울 수 있으며, 테스트베드 개발자는 개발 일정과 개발 환경의 특수성 등으로 인해 테스트베드 사용자를 면밀히 고민하기 어려워 다양한 사용자 요구사항을 수용할 수 있는 주문형 가상 네트워크의 구성 능력이 부족할 수 있다. 결국, 이들 4가지 특성을 모두 만족할 수 있는 미래인터넷 테스트베드 플랫폼 구성은 쉽지 않은 작업일 것이다.

Ⅱ. 네트워크 가상화의 기술 동향

지금까지 발표된 미래인터넷 테스트베드 플랫폼 관련 논문들을 살펴보면 네트워크 가상화를 세부적으로 두 가지 측면에서 다루고 있다. 즉, 호스트 가상화와 링크 가상화가 바로 그것들인데, 호스트 가상화는 VMM과 가상머신을 의미하고, 링크 가상화는 가상 네트워크 인터페이스를 의미한다. 본 고에서는 여기에 세 가지를 더 추가하는데, 라우팅 가상화와 가상머신 마이그레이션, 그리고 가상 네트워크 아키텍처이다. 라우팅 가상화는 현재의 인터넷 인프라에서 선도적인 제품 가운데 극소수가 이 기능을 지원하고 있으나, 개념 자체가 미래인터넷에서 추구하고자 하는 것을 부분적으로 수용하고 있으며, 이는 호스트 가상화와 링크 가상화와는 다른 내용이기 때문에 네트워크 가상화의 세부 기술로 포함하였다. 가상머신 마이그레이션은 부분적으로 VMM에서도 수용할 수 있으나, 시스템 차원에서 접근하는 것이 옳다고 판단되어 네트워크 가상화의 세부 기술로 독립시켰다. 그리고 기존의 네트워크 아키텍처만으로 가상 네트워크의 제어 및 관리를 지원할 수 없기 때문에 기존의 데이터, 제어 그리고 관리 평면에 가상화 평면을 추가하는 가상 네트워크 아키텍처가 요구되어 이를 또 하나의 세부 기술로 포함하였다.

참고로, 네트워크 가상화 기술로 VLAN 또는 VPN 등을 포함할 수 있지만, 이들 기술은 특정 계층의 트래픽 격리에 적용할 수 있는 기술로서, 현재의 네트워크 인프라에서 잘 활용되고 있기 때문에 미래인터넷의 네트워크 가상화 기술로 포함하지 않는 것이 적절하다.

1. 호스트 가상화

하드웨어 위의 시스템 OS를 포함한 가상화 계층을(좁은 의미로는 시스템 OS를 제외한 가상화 계층만을 대상으로 함) VMM이라고 말할 수 있는데, 이를 일명 하이퍼바이저라고도 한다. 따라서, VMM은 하나의 호스트에서 다양한 OS 및 관련 응용(가상머신)을 실행 가능케 하는 하드웨어와 가상머신 사이의 소프트웨어 계층이다. 참고로, 가상머신은 자신의 OS 이외에 실행 프로그램, 파일 시스템, 사용자(root 포함), 네트워크 인터페이스(IP 주소 포함), 라우팅 테이블, firewall 규칙 등을 포함한다. 가상머신은 일반적으로 가상화 영역에서 용어이며, 미래인터넷 테스트베드에서는 슬리버라는 용어로 사용하고 있다.

[4]에 따르면 시스템 가상화(여기서는 호스트 가상화)를 Native VM 시스템, Hosted VM 시스템, 그리고 OS 확장 VM 시스템으로 분류한다. 여기서 Native VM 시스템과 Hosted VM 시스템은 각각이 동일하게 전가상화(full-virtualization), 반가상화(para-virtualization), 그리고 하드웨어 지원 가상화로 세분하고, OS 확장 VM 시스템은 커널 가상화와 OS 기반 가상화로 세분한다. 참고로, 위키피디아에 따르면, Native VM 시스템은 하드웨어를 제어하고 Guest OS를 모니터링하기 위해 VMM이 하드웨어 위에서 바로 동작하고, Hosted VM 시스템은 VMM이 전통적인 OS 환경 내에서 동작한다.

〈표 1〉 Native VM 시스템의 주요 특성 비교

	Citrix XenServer 5.0	MS Hyper-V	Oracle VM 2.1	Red-Hat RHEL 5.2	Sun xVM Server 1.0	VMware ESX 3.5
VMM	Xen	Proprietary	Xen	Xen	Xen	Proprietary
Service OS	Linux	Windows	Linux	Linux	Solaris	Linux
VMs(max)	75	192	unlimited	unlimited	40	80
CPU 가상화 (하드웨어 지원)	AMD-V SVM, Intel VT-x	AMD-V SVM, Intel VT-x	AMD-V SVM, Intel VT-x	AMD-V SVM, Intel VT-x	AMD-V SVM, Intel VT-x	AMD-V SVM, Intel VT-x
메모리 가상화 (하드웨어 지원)	AMD-V RVI		AMD-V RVI	AMD-V RVI	?	AMD-V RVI
I/O 가상화 (하드웨어 지원)		Intel VT-c	Intel VT-c	?	?	Intel VT-c
vSMP(max)	8	4	32	8	32	4
vNICs(max)	7	12	8	?	100	4

ESX, Hyper-V 등을 포함하는 Native VM 시스템(Citrix XenServer, Oracle VM, Red-Hat RHEL, Sun xVM Server 등)의 주요 특성들에 대한 개략적인 비교는 <표 1>과 같다[5].

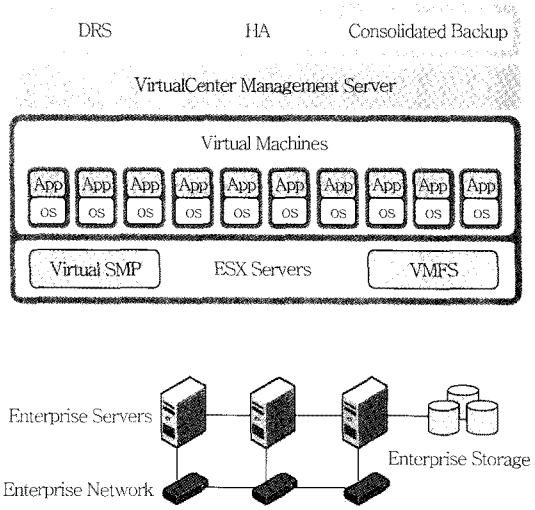
본 고에서는 현재까지 미래인터넷과 관련하여 분석한 자료를 기초로 좀더 간략히 접근하기 위해서 Hosted VM 시스템은 제외하고, Native VM 시스템에서 하드웨어 지원 가상화를 제외한다. 따라서, 잠정적으로 호스트 가상화를 전가상화, 반가상화, 그리고 OS 기반 가상화로만 분류한다.

가. 전가상화

전가상화는 이진코드 변환 기법(binary code translation)을 사용하는 VMM 위에서 서로 다른 Guest OS(Linux, Windows, Solaris, Netware 등)를 갖는 다수의 가상머신을 실행하는 구조로서, Guest OS를 수정할 필요가 없다. 하지만, 하드웨어 에뮬레이션으로 인한 성능 저하가 단점이다. 이러한 방식의 가상화 유형으로 VMware ESX/ESXi 등이 있다.

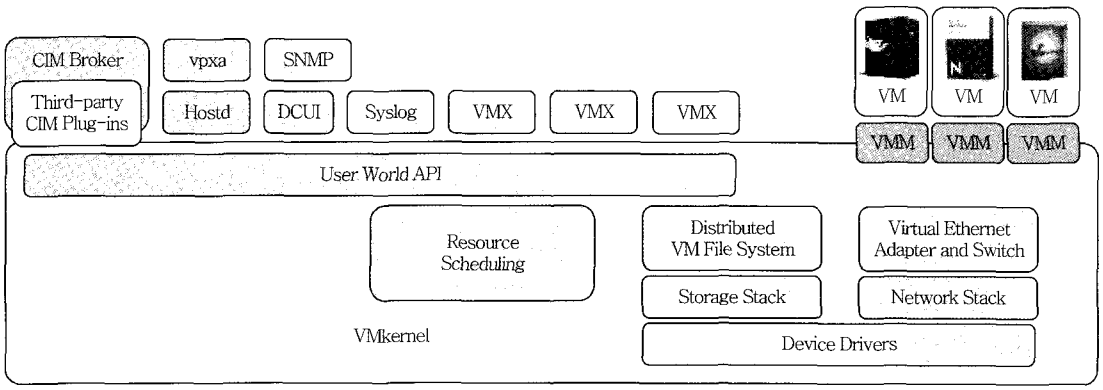
• VMware ESXi

VMware사는 전가상화 및 반가상화 등 비교적 폭넓은 VMM을 지원하고 있는데, (그림 2)와 같이 노드 내부의 가상화뿐만 아니라 자원 관리, 라이브 마이그레이션, 가상 관리 서버 등 서버 차원에서 가상화 인프라를 제공하고 있다. 가상화 인프라의 주



(그림 2) VMware 인프라 구조

요 구성 요소로는 ESX 서버, 가상머신 파일 시스템(VMFS), 가상 SMP, 가상 센터 관리 서버, VMotion, 분산 자원 스케줄러(DRS) 등을 포함하고 있다. VMware의 VMM은 CPU, 메모리, 그리고 I/O(NIC, 디스크)에 대한 가상화를 지원하고 있다. CPU 가상화를 위해 직접 실행(direct execution)과 이진 코드 번역 기술을 사용하며, 메모리 관리를 위해 가상머신 OS의 가상 메모리 테이블과 물리적 메모리 테이블 간의 대응을 표시하는 페이지 테이블(shadow page table)을 사용한다. 그리고 I/O 가상화를 위해 고성능의 I/O 디바이스 드라이버는 직접 VMM에 위치시키고, 그 이외의 I/O 디바이스 드라이버는 특정 도메인



(그림 3) VMware ESXi 구조

(일명 서비스 콘솔)에 위치시켰다[6].

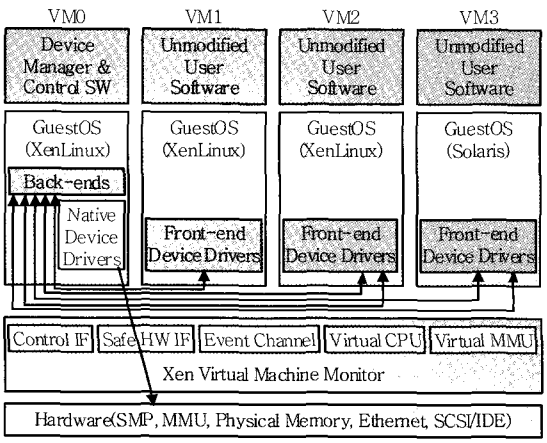
ESXi는 기능성, 성능, 그리고 확장성 측면에서 ESX와 동일하지만, ESXi에서는 서비스 콘솔을 제거하고, 이와 관련하여 부팅 방식, 사용자 관리 인터페이스, 하드웨어 수준 관리 등을 보완하였다고 한다. (그림 3)과 같이 ESXi의 주요 컴포넌트로는 VM 커널(VMM), 직접 콘솔 사용자 인터페이스(DCUI), 관리 에이전트, 공통 정보 모델(CIM) 등이 있다. VM 커널은 POSIX 계열의 OS로 VMware에서 자체 개발하였으며, 일반 OS 기능(프로세스 생성 및 제어, 파일 시스템 등)에 가상화 관련 기능과 자원 스케줄링, I/O 스택, 디바이스 드라이버 기능 등을 확장하였다. DCUI는 사용자 인터페이스를 서비스 콘솔을 통한 간접 인터페이스 방식에서 BIOS와 유사한 메뉴 기반 인터페이스 방식으로 수정한 것으로, 암호 설정, 네트워크 구성, 로깅, 네트워크 시험, 에이전트 재시작, 그리고 디폴트 복구 등을 처리한다. 그리고 CIM은 컴퓨팅 자원의 표현과 관리를 위해 표준 API를 이용한 하드웨어 수준의 관리 시스템을 제공한다. 그리고 사용자 관리 기능을 수행하는 hostd, 가상 센터로의 접속을 지원하는 에이전트 기능을 수행하는 vpxa 등이 있다[7].

나. 반가상화

VMM 위에서 서로 다른 Guest OS(Linux, Windows, Solaris 등)를 갖는 다수의 가상머신을 실행하는 구조로서, 전가상화의 하드웨어 에뮬레이션으로 인한 성능 저하의 단점을 보완하였으나, 하드웨어 API를 직접 Guest OS에 반영하기 위해 Guest OS를 수정해야 한다. 이러한 방식의 가상화 유형으로 Xen 등이 있다.

• Xen

캠브리지 대학의 연구 프로젝트로 시작한 Xen은 주로 x86 계열의 프로세서를 위한 VMM으로, GNU GPL 규정을 따르는 공개 소프트웨어다. 2003년 Xen 1.0 공개 이후, Xen 2.0 및 Xen 3.0에 이르고 있으며, Xen에서 패킷 처리 흐름은 다음과 같다. 즉, 외부에서 입력되는 패킷은 Dom0의 디바이스 드라이버를 통해 패킷을 수신하고, 이를 back-end 드라이버를 통해 큐로 구현된 가상 인터페이스로 전달한다. 이 가상 인터페이스는 가상머신별로 존재하기 때문에 해당 가상머신에게만 전달된다. 가상머신이 패



(그림 4) Xen 2.0 구조

킷을 전송하는 경우도 유사하게 가상머신이 가상 인터페이스에 패킷을 삽입하면 Dom0의 back-end 드라이버를 통해 이를 받고 디바이스 드라이버를 통해 외부로 전송한다. (그림 4)는 이러한 Xen 2.0 구조를 나타낸다[8].

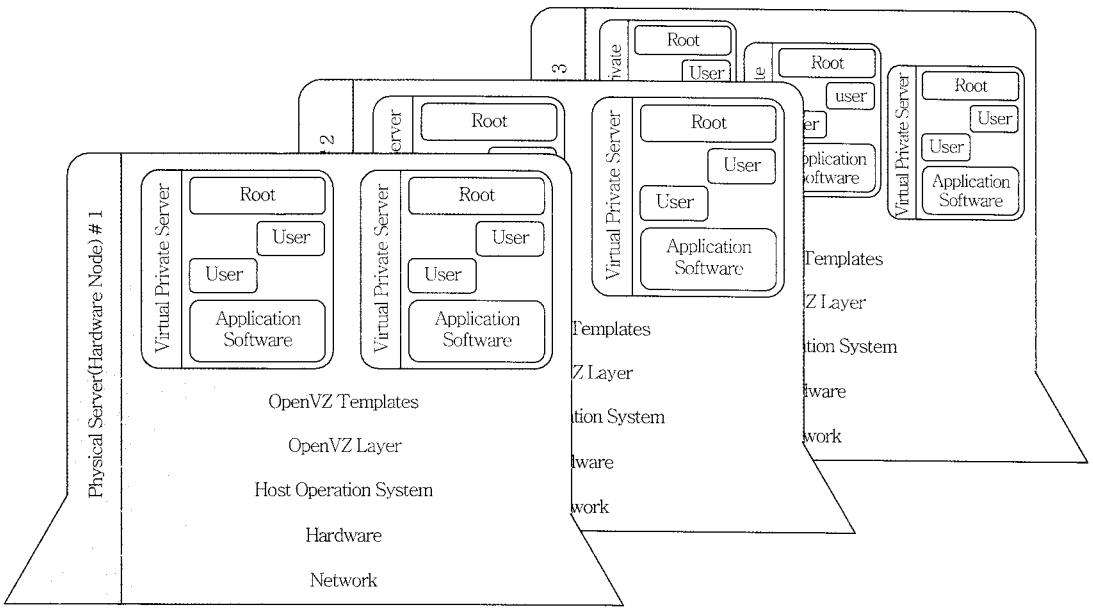
Xen에서는 부팅시 제어 인터페이스(control IF)를 사용할 수 있는 권한을 가진 특수 도메인(VM0 또는 Dom0)을 하나 생성한다. 이 도메인은 Guest OS의 초기 구조를 구성하는 등 호스트 가상화를 위한 독점적인 제어 및 관리 기능을 제공한다. 그리고 제어 인터페이스는 일반 도메인의 생성 및 삭제, 도메인 스케줄링 파라미터, 메모리 할당 그리고 디스크 및 네트워크 디바이스 접근 등에 대한 제어를 수행한다. 현재, Xen 2.0의 경우, Dom0 OS로 Linux 2.4/2.6, NetBSD 3.0이 가능하며, DomU(VM0 이외의 VM) OS로 Dom0 OS들과 NetBSD 2.0, Plan9, FreeBSD 5가 가능하다. Xen 3.0의 경우, Dom0 OS로 Linux 2.6, NetBSD 4.0 베타 버전이 가능하며, DomU OS로 Dom0 OS와 NetBSD 3.1, Solaris 10이 가능하다. 참고로, Xen 3.0에서는 그래픽, SMP Guest OS, 인텔 VT-x 등을 위한 사항들이 보완되었다[9].

다. OS 기반 가상화

OS 기반 가상화는 하나의 시스템 OS 커널 위에서 다수의 가상머신을 실행하는 구조로서, 성능과 지원 가상머신의 개수 측면에서 다른 가상화 방식에 비해 장점이 있다. 하지만, 시스템 OS와 Guest OS가 동일해야 하며, 시스템 OS의 장애가 가상머신들에게 영향을 줄 수 있다. 이러한 방식의 가상화 유형으로 OpenVZ, Linux-VServer 등이 있다.

• OpenVZ

OpenVZ는 가상화 및 자동화 관련 소프트웨어를 다루는 Paralleles사의 지원으로 수행되고 있는 프로젝트로서, 역시 GNU GPL 규정을 따르는 공개 소프트웨어다. OpenVZ는 스케줄링 등과 같은 기본적인 Linux 커널 위에서 가상머신의 격리, 자원(CPU, RAM, 디스크 공간 등) 관리, 마이그레이션 등과 같은 가상화 기능들을 추가하였다. 이 OpenVZ는 현재 Fedora Core 3/4, Red Hat Enterprise Linux 4에서 설치 가능하며, 가상머신들은 서로 다른 버전의 Linux를 사용할 수 있다. OpenVZ 위에 가상머신의 운용에 필요한 동작을 지원하기 위해 사용하는 툴(vzpkg)과 메타 데이터가 있는데 이를 템플릿이라



(그림 5) 물리적 노드 단위의 OpenVZ 구조

고 한다. 그리고 모든 자원은 런타임에도 변경될 수가 있다고 한다. (그림 5)는 물리적인 노드 단위의 OpenVZ 구조를 나타낸다.

가상머신의 격리를 지원하기 위해, 각 가상머신은 자신의 파일, 사용자(root 포함) 및 그룹, 프로세스 트리, 네트워크(IP 주소, 라우팅 규칙 등), 디바이스, IPC(공유 메모리, 세마포워 등) 등을 유지하고 있다. 디스크 관리는 2-수준으로 진행되는데, 1-수준에서는 OpenVZ 관리자에 의해 가상머신 단위로 디스크 공간과 inode 수를 설정하고, 2-수준에서는 해당 가상머신의 root에 의해 표준 툴을 사용하여 사용자 또는 그룹 단위로 디스크 공간을 설정한다. CPU 스케줄링 역시 2-수준으로 진행되는데, 가상머신의 우선순위와 임계치 등을 고려하여 CPU를 점유할 해당 가상머신을 먼저 결정한 후, 표준 툴을 사용하여 가상머신 내의 프로세스를 결정한다.

마이그레이션 과정을 살펴보면, 일단 마이그레이션 전에 가상머신을 중지하고, 관련 상태 정보를 디스크에 저장한다. 이후, 이 상태 정보를 지정 호스트로 전달하고 해당 가상머신을 기동한다. 이 과정 중에서 필요하면 네트워크 재설정을 수행하는데, 전체적으로 약 수 초가 걸린다고 한다[10],[11].

• Linux VServer

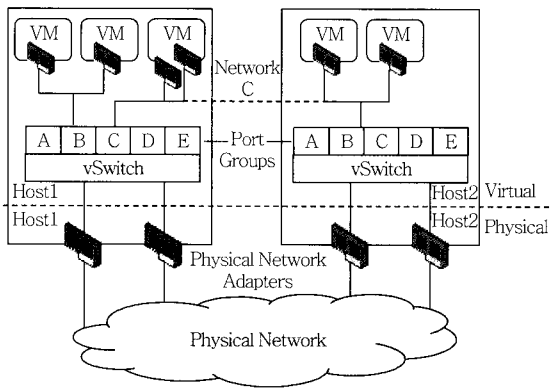
Linux VServer는 2003년경, 하나의 Linux 서버를 모니터링, 백업, UPS, 하드웨어 구성 등의 공통 태스크를 공유하면서 다수의 가상 서버로 분리하기 위해 시작한 개인 프로젝트가 Linux 커뮤니티 프로젝트로 발전한 것으로, 역시 GNU GPL 규정을 따르는 공개 소프트웨어다. 이러한 Linux VServer는 하드웨어, 커널, 그리고 가상머신의 3개의 컴포넌트로 구성되어 있으며, OpenVZ와 같이 하나의 물리적 서버 위에서 생성 가능한 가상머신의 수에 제한이 거의 없다. 하지만, 가상머신의 격리 이외의 가상화 기능(자원 관리, 마이그레이션 등)은 지원하지 않는다. 현재 미래인터넷 테스트베드 중의 하나인 PlanetLab에서 사용하고 있다[12].

2. 링크 가상화

VMware 같은 경우에 I/O 가상화에서 디스크 및 네트워크 디바이스에 대한 가상화를 다루고 있으나, 의외로 다수의 논문이나 WP에서 링크 가상화를 주제로 많이 다루고 있었다. 여기서 말하는 링크 가상화라 함은 하나의 물리적인 네트워크 디바이스(예: 10G 이더넷 디바이스)에서 다수의 가상 네트워크 인터페이스(VNIC) 기능을 지원해 주는 기술을 말한다.

• VMware

VMware에서 가상머신들은 하나 또는 그 이상의 자신의 IP 주소와 가상 네트워크 인터페이스를 가지고 있다. 그리고 가상 네트워크 인터페이스는 표준 이더넷 프로토콜에 따라 동작한다. 가상 네트워크 인터페이스를 물리적 네트워크 인터페이스를 접속시키기 위해 포트 그룹이라는 가상 네트워크 인터페이스에 대한 다중화 개념을 적용한다. 이를 통해 네트워크 보안, 네트워크 분할, 성능, 그리고 트래픽 관리 등 제반 네트워크 정책을 설정할 수 있다. 최종적으로 가상 스위치는 포트 그룹과 물리적 네트워크 인터페이스 사이에서 트래픽 스위칭 역할을 수행한다. (그림 6)은 이러한 VMware의 링크 가상화를 나타낸다[6].

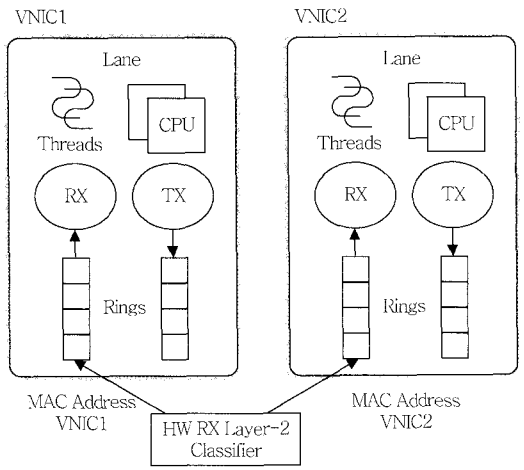


(그림 6) VMware 네트워킹 구조

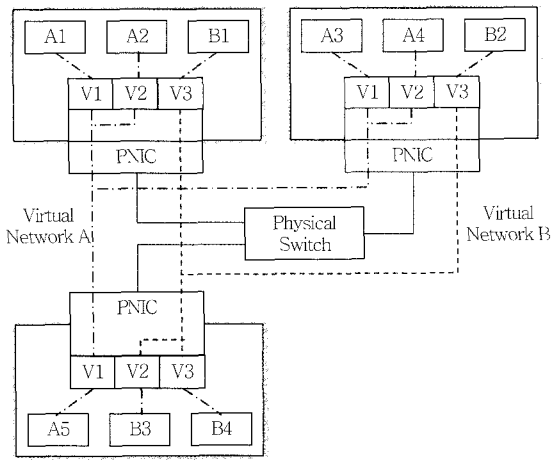
• Crossbow

OpenSolaris 기반 Crossbow는 MAC 계층의 자원(송수신 큐・버퍼, DMA, CPU, 전용 소프트웨어

자원 등)을 가상머신 단위로 할당하여 트래픽을 분리할 수 있는 가상 네트워크 인터페이스 기술을 지원한다. (그림 7)은 Crossbow의 링크 가상화를 위한 하드웨어 및 소프트웨어 자원들을 나타내는데, Crossbow에서는 자원 격리를 위한 물리적 네트워크 인터페이스 카드의 하드웨어 자원이 더 이상 존재하지 않을 경우에는 자원을 공유하는 소프트웨어 기반 링크 가상화 방식이 사용될 수 있다고 한다. 또한, 하나의 물리적 노드 위에서 가상머신, VNI, 가상 스위치, 이더스터브 등을 하나의 박스 형태로 구성하는 가상 와이어 기능을 지원할 수 있다고 한다. 이러한 가상 와이어 특성을 활용하여 공중 네트워크와



(그림 7) 링크 가상화 자원



(그림 8) 네트워크 기반 가상 와이어

분리하여 물리적 토폴로지와 관계 없이 다양한 실험을 진행할 수 있다. 그리고 VLAN 태깅과 연계하여 다수의 물리적 노드 위에서 가상 와이어 범위를 확장할 수 있다. (그림 8)은 네트워크 수준에서의 가상 와이어 구성에 대한 하나의 예를 나타낸다[13].

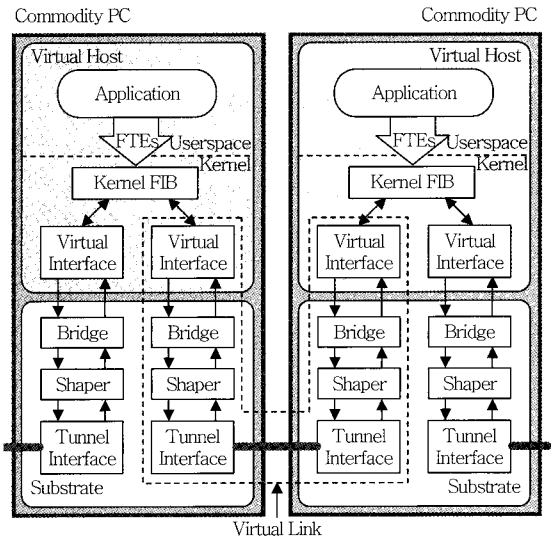
• Cisco UCS M81KR 가상 네트워크 인터페이스 카드

현재, 데이터 센터의 인프라는 트랜잭션의 증가와 가상화 기술의 적용 확대 추세에 있다. 이는 각 서버에서 I/O 증대를 야기시켰고, 동적 I/O 관리의 필요성을 제기하였다. Cisco는 이러한 요구사항에 대응하기 위해 UCS M81KR이라는 가상 네트워크 인터페이스 카드를 개발하였는데, 이 카드는 Cisco UCS B 시리즈의 블레이드 서버 상에서 Ethernet 및 FCoE 환경에서 최적화된 메자닌 카드이다.

이 카드는 듀얼 포트의 10GE에서 최대 128개의 PCIe 기반 가상 인터페이스를 지원한다. 그리고 NIC와 HBA를 동적으로 구성할 수 있으며, VMware의 가상 데이터 센터를 통해 가상머신 단위로 네트워크 정책 및 관리 정보를 제어한다. 이 카드를 사용함으로써 서버 구성 및 운영의 단순화, 스위치 및 어댑터에 대한 CAPEX와 OPEX의 절감, 그리고 전력 비용의 감소 효과를 얻을 수 있다고 한다[14].

• Trellis의 링크 가상화

Trellis는 링크 가상화를 위해 해당 토폴로지 상에 있는 가상 노드들과 패킷 교환을 위해 GRE 터널링 메커니즘을 사용한다. 이 메커니즘을 통해 각 가상 네트워크는 가상 노드 네트워크 내에서만 식별 가능하다면 동일한 IP 주소를 서로 중복하여 사용할 수 있으며, non-IP 패킷을 포워딩 할 수 있다고 한다. 가상 노드 내의 가상 인터페이스와 서브스트레이트의 터널 인터페이스의 이더넷 기반 대응 관계를 통해 점대점(short bridge 이용) 및 점대다중점(bridge 이용) 가상 링크를 구성하여 가상 노드들 간 통신 환경을 설정한다. (그림 9)는 가상 인터페이스와 터널 인터페이스를 포함하는 Trellis의 구조를 나타낸다[15].



(그림 9) Trellis 구조

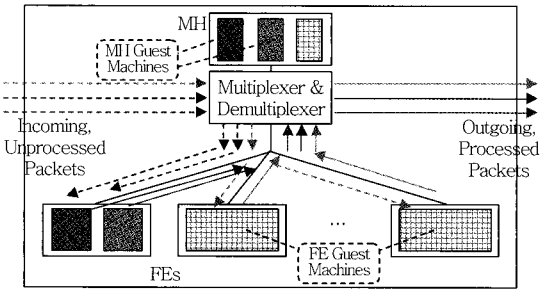
지금까지 언급된 링크 가상화에서 Trellis를 제외한 대부분의 경우에 가상 네트워크 인터페이스에 해당되는 가상 MAC 주소를 이용하여 상대 노드와 직접 ARP 패킷을 교환하는 것으로 보인다.

3. 라우팅 가상화

순수한 의미에서 라우팅 가상화는 하나의 물리적인 라우터에서 자원을 엄격히 분리하여 다수의 가상 라우터를 구성하는 기술로, 하드웨어 자원, 제어 평면, RIB 및 FIB로 구성된 하나의 가상 라우터 인스턴스가 각각 독립되어 동작한다. 하지만, 현재로서는 각 플랫폼의 특성에 따라 제한적인 라우팅 가상화 기능을 지원하고 있다.

• PdP

PdP 노드는 (그림 10)과 같이 패킷 처리를 병렬로 처리하는 PC 머신들의 클러스터로써, OS 수준의 가상화 기술을 기반으로 하나의 관리 호스트 머신과 다수의 포워딩 엔진 머신, 그리고 다중화/역다중화 머신으로 구성한다. 관리 호스트의 게스트 머신에서 제어 평면의 기능을 실행하고, 포워딩 엔진 머신에서는 주소 룩업과 트래픽 셰이핑 등과 같은 패킷 처리 기능을 수행한다. 그리고 다중화/역다중화 머신



(그림 10) PdP 노드 구조

은 입력 패킷을 특정 포워딩 엔진으로 할당하고(패킷 분류기의 기능) 출력 패킷을 특정 인터페이스로 지정한다(패킷 디스패처기의 기능). 만일 포워딩 엔진의 성능을 개선하고자 한다면 다중화/역다중화 머신과 포워딩 엔진 머신을 저가의 상용품(예: PC) 대신 네트워크 프로세서나 NetFPGA를 사용할 수 있다고 한다.

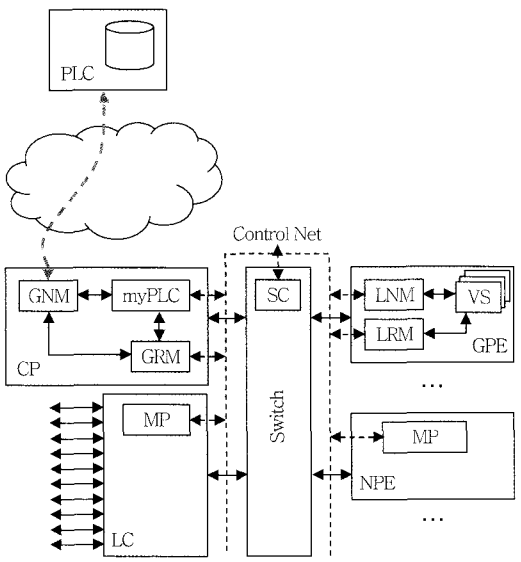
PdP에서 기술적인 이슈는 패킷 처리를 병렬화함으로써 발생할 수 있는 out-of-order 패킷 문제이다. 만일, 하나의 가상 네트워크에 포함되는 다수의 포워딩 엔진의 게스트 머신을 다수의 물리적 머신에 분산 배치하였을 때 out-of-order 패킷이 발생할 수 있다. 따라서, out-of-order 패킷 문제를 발생시키지 않는 근본적인 처방은 각 물리적 머신에 하나의 가상 네트워크를 할당하는 것이지만, 이는 자원의 효과적 활용에 장애 요소가 될 수 있다[16].

• SPP

SPP는 일반 PC에서 패킷 처리를 수행하였을 때의 성능 문제를 해결하기 위해 고속 패킷 처리를 인텔 IXP 2850 네트워크 프로세서에서 수행하고, 제어 평면 기능을 일반 프로세서에서 수행한다. 그리고 이 플랫폼은 현재 미국, 유럽 등에서 널리 사용되는 프린스톤 대학의 PlanetLab 제어 프레임워크를 채용하고 있다. (그림 11)은 GNM 및 GRM으로 이루어진 제어 프로세서(control processor), LNM 및 LRM 그리고 각각의 가상 서버(슬라이스)로 이루어지는 GPE, 그리고 고속의 패킷 처리를 수행하는 NPE로 구성되는 SPP의 제어 구조를 보여주고 있다. NPE를 사용하지 않는 slow path 실험과 NPE를 사용하는

fast path 실험을 동시에 지원한다. Fast path 실험의 경우, 사용자는 NPE 코드 특성, 필터 테이블 크기, 큐, 버퍼, SRAM, 대역폭 등의 자원 파라미터 등을 제시해야 하며, GRM은 적절한 NPE를 선정하고, LRM으로 해당 슬라이스의 초기 구성을 요청한다. 이 플랫폼은 Linux VServer 기반의 PlanetLab 제어 프레임워크를 적용한 것과 인텔 IXP 2850 네트워크 프로세서를 사용하여 포워딩 엔진의 파이프라인화를 통해 패킷을 고속으로 처리하는 것이 특징이다.

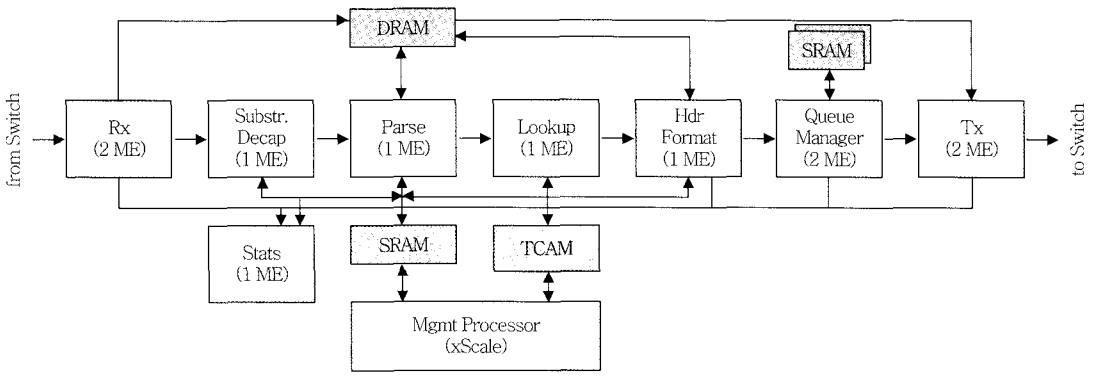
좀더 세부적으로 살펴 보면, 16개의 마이크로 엔진(ME) 내의 각 모듈은 8개 패킷을 동시에 병렬 처리하며, 모듈들은 하나 또는 두 개의 ME를 할당 받아 패킷 처리 과정에 따라 순서적으로 패킷을 제어한다. 좀더 구체적으로 살펴 보면, 수신 모듈은 스위치로부터 수신된 패킷을 DRAM에 복사하고, 패킷 포인터를 전달한다. 서브스트레이트 제거 모듈은 SRAM 룩업을 통해 패킷의 슬라이스를 결정한다. 파싱 모듈 SRAM에 저장된 슬라이스별 코드와 슬라이스 종속적인 정보(헤더 등)를 확인하여 룩업-키를 구성한다. 룩업 모듈은 관리 프로세서의 지원을 통해 해당 슬라이스에서 정의한 패킷 필터링을 처리하고, 112비트 스트링의 룩업-키, 슬라이스 식별자를 이용하여 TCAM 룩업을 수행한다. 헤더 포맷 모듈은 해당 슬라이스에 따라 패킷 헤더를 수정하고, 큐 관리자 모듈은 패킷 스케줄링 정책에 따라 패킷을 슬라이스별로 큐에 할당한다. 송신 모듈은 해당 패킷을 스위치로 포워딩 한다. 그리고 통계 모듈은 슬라이스별 다양한 통계 및 성능 정보를 계산하여 SRAM에 저장하고, 관리 프로세서는 이 정보에 접근한다. (그림 12)는 SPP의 네트워크 프로세서에서 각각의 패킷 처리 모듈의 ME 할당 방법 그리고 패킷 처리 과정을 보여주고 있다[17].



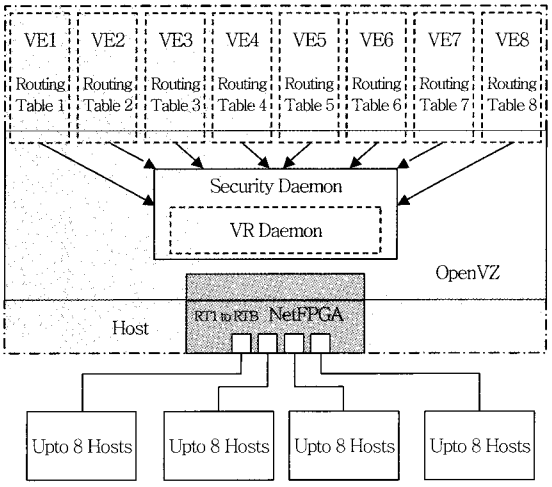
(그림 11) SPP 제어 구조

• NetFPGA 기반 가상 라우터

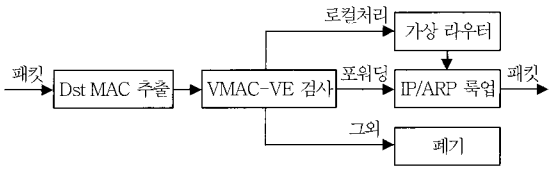
NetFPGA는 FPGA 기반의 패킷 처리 플랫폼으로, 4개의 1GE 이더넷 인터페이스를 가지고 있다. 하나의 가상 라우터는 NetFPGA의 1GE 이더넷 인터페이스 수만큼인 4개의 가상 네트워크 인터페이스를 포함할 수 있으며, 온칩 메모리의 크기에 따라



(그림 12) SPP 패킷 처리 과정



(그림 13) 조지아텍의 가상 라우터 개념 구조

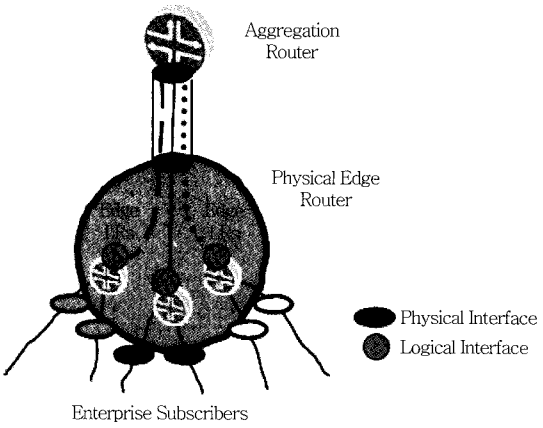


(그림 14) 조지아텍 호스트의 내부 패킷 제어 과정

확장 가능하다고 한다. NetFPGA를 활용한 조지아텍의 호스트는 OpenVZ를 활용하여 최대 8개의 가상 라우터를 생성할 수 있고, 현재로서는 하나의 물리적 1GE 이더넷 인터페이스는 최대 8개의 호스트와 접속할 수 있다고 한다. (그림 13)은 이러한 조지아텍 호스트의 가상 라우터 개념 구조를 나타낸다. 가상 라우팅 데몬 내의 가상 라우팅 룩업 모듈은 VMAC-VE 대응 테이블과 해당 가상 라우터에 대한 MAC 주소를 유지하고 있는 콘텍스트 레지스터의 제어 정보를 이용하여 송수신 패킷을 제어한다. (그림 14)는 조지아텍 호스트에서 패킷 수신 후 다시 전송할 때까지의 내부 패킷 제어 과정을 나타낸다[18].

• Juniper

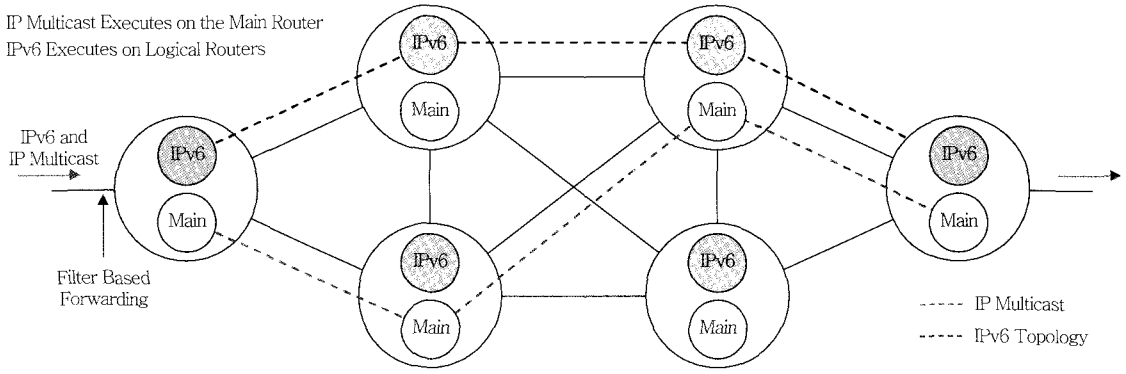
가상 라우팅은 미래인터넷이 논의되기 전부터 상용 기능이 출시되었기 때문에 미래인터넷의 이슈가 아니다. 하지만, 네트워크 가상화를 계층적으로 접근할 때 계층 3에 대한 가상화를 언급하지 않을 수



(그림 15) Juniper의 논리적 라우터

없는데, 이 부분이 바로 가상 라우팅이다. 서비스 제공자의 기업형 가입자가 원하는 라우팅 구성을 지원하기 위해 물리적 라우터를 추가로 배치한다는 것은 CAPEX 및 OPEX 비용을 고려할 때 올바른 접근 방법이 아니다. 이를 해결하기 위해, 하나의 물리적 라우터에서 다수의 가상 라우터를 구성하고, 각각의 가상 라우터가 자신만의 RIB 및 FIB를 생성·관리하게 하는 가상 라우팅 방법을 구현하게 되었다. 이러한 방법을 통해 물리적 라우터를 추가로 구성하지 않고, 네트워크 통합, VPN 서비스, 응용별 라우팅, 그리고 서비스 개념 증명 등을 효과적으로 수행할 수 있다.

구체적인 사례 중의 하나로 Juniper의 경우, (그림 15)와 같이 하나의 물리적 에지 라우터에서 다수의 논리적 에지 라우터(라인카드 수준이 아닌 논리적 인터페이스 수준이며, 가상 라우터와 약간의 차이가 있다고 함) 기능을 통해 애그리게이션 라우터로 접속한다. 이러한 기능은 M 및 T 시리즈 등의 JUNOS 상에서 하나의 물리적 라우터에서 최대 16개의 논리적 라우터를 구성할 수 있다고 한다. 이러한 논리적 라우터를 이용하여 동일한 사용 플랫폼 위에서 서비스 제공자는 새로운 서비스를 도입하기 전에 상용 라우터 위에서 IP 멀티캐스팅 또는 IPv6와 같은 서비스의 개념 검증을 완료한 후 최종 도입 여부를 결정할 수 있다. (그림 16)은 이러한 예시를 나타낸다[19].



(그림 16) IPv6 및 IP 멀티캐스트 개념 검증

4. 가상머신 마이그레이션

데이터 센터의 서버가 아닌 네트워크 내부에서 안정성이 일차적으로 요구되는 상황에서 가상머신 마이그레이션이 필요할 것인가에 대한 확신은 아직 없지만, 향후 virtualized NAT 및 DNS 등 다양성 네트워크 응용의 가능성을 염두에 두면 가상머신 마이그레이션이 유용한 기술이 될 수 있는 여지는 충분히 있다고 생각한다.

• Xen

Xen의 마이그레이션 전략은 마이그레이션 사전준비(pre-migration), 자원 예약(reservation), VM 메모리 복사(iterative pre-copy), 이전 VM 중단(stop and copy), 마이그레이션 완료 확인(commitment), 그리고 신규 VM 활성(activation) 등의 여섯 단계로 이루어지며, 좀더 세부적으로는 다음과 같이 진행된다[20].

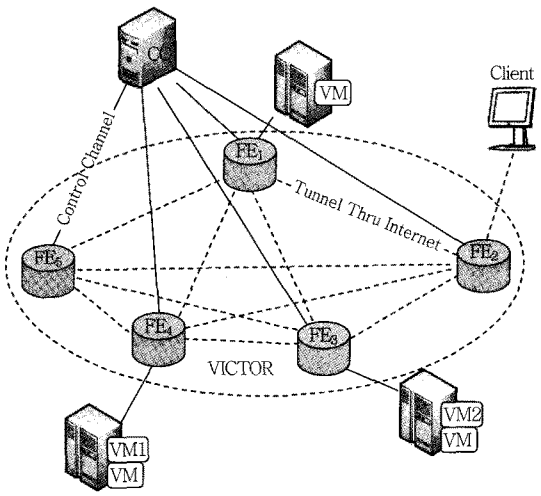
① 마이그레이션 사전준비: 해당 VM의 자원 할당이 가능한 목표 호스트를 미리 지정
② 자원 예약: VM 마이그레이션 신호를 발생하고, 해당 VM에 대한 초기 구성 작업(VM 크기 및 자원 확인 등)을 수행
③ VM 메모리 복사: 이전 호스트에서 신규 호스트로 VM 메모리를 복사하고, 이전 호스트에서 이전 복사 과정중 메모리 변경이 일어난 페이지를 반복적으로 복사
④ 이전 VM 중단: 이전 호스트의 VM을 중단하고, 신규 호스트로 CPU 상태와 VM 메모리의 최종 변경을 반영
⑤ 마이그레이션 완료 확인: 이전 및 신규 호스트 간 VM 마이그레이션의 완료 상황을 확인
⑥ 신규 VM 활성: 신규 호스트의 VM을 동작시키고, 내외부 네트워크의 연결(디바이스 드라이버 접속 및 IP 주소 광고 등)

• GEC3 OpenFlow 데모

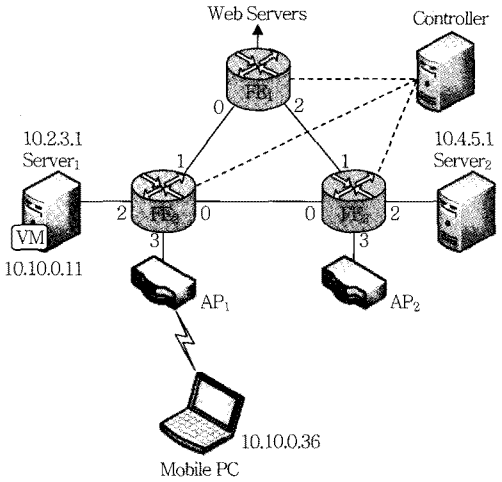
SIGCOMM 2008에서 최고 데모상을 수상한 것으로 라우터, 스위치 그리고 WiFi AP와 같은 기존 장비(Cisco Catalyst 6500, Juniper MX, HP ProCurve 5400, NEC IP 8800 등)에 탑재된 OpenFlow 기능을 사용하여 게임을 하는 게임 모바일 사용자가 이동하더라도 IP 주소를 변경하지 않고 새로운 게임 서버로 모바일 VM의 마이그레이션을 통해 서비스 연속성을 지원한다. OpenFlow Controller는 노드의 요청에 따라 10 튜플의 OpenFlow 포워딩 테이블을 제어하고, 이동 사용자의 근접 위치에 있는 게임 서버로 VM을 이동시킨다. 원래, OpenFlow의 사용 목적은 기존 상용 장비에서 최소한의 기능 추가를 통해 새로운 아이디어를 실험할 수 있게 하는 것으로, VM 마이그레이션 과정에서 패킷 라우팅의 변경에 효과적으로 사용할 수 있다는 가능성을 보여 주었다.

• VICTOR

클라우드 컴퓨팅 분야에서 데이터 센터간의 서



(그림 17) VICTOR 구조



(그림 18) VICTOR 시험망

비스 마이그레이션을 지원하기 위해 네트워크 아키텍처를 제시한 것으로, 이 개념을 검증하기 위해 OpenFlow를 확장(모바일 노드 등록 및 L3 라우팅 부분)하여 프로토타이핑 하였다. 네트워크를 마치 소프트 라우터와 같이 제어 평면과 데이터 평면으로 분리한다. VICTOR는 네트워크를 제어 평면의 기능은 중앙 제어기에서 수행하고, 데이터 평면의 포워딩 기능은 각 노드에 분산되어 있는 하나의 커다란 가상 라우터로 간주한다. 그리고 특정 서비스를 실행하는 VM 서버는 지정된 인접 노드에 접속되어 있다고 본다. (그림 17)은 이러한 VICTOR 구조를 보여주고 있다.

하나의 데이터 센터 내부에서의 VM 마이그레이션 가운데 특정 서브넷 내의 VM 마이그레이션은 OpenVZ 등과 같은 호스트 가상화에서 지원되기 때문에 문제가 되지 않으나, 서브넷간 마이그레이션은 VM 서버와 노드간 ARP 프로토콜과 노드와 중앙 제어기간 새로운 위치 정보의 등록 및 관련 포워딩 테이블의 재구성이 필요하다고 한다.

데이터 센터간 VM 마이그레이션의 경우, 각 모바일 VM은 자신의 IP 주소를 할당받아야 하며, 데이터 센터들간의 보더 라우터들은 이들 IP 정보를 광고한다. VM 마이그레이션이 발생할 경우, VM 서버와 노드간 ARP 프로토콜과 노드와 중앙 제어기간 새로운 위치 정보의 등록 및 관련 포워딩 테이블의 재구성에 추가하여 보더 라우터들 역시 포워딩 테이블을 재구성해야 한다. 이후, 데이터 센터간 사용자와 새로이 설정된 터널을 통해 트래픽을 교환하게 된다. VICTOR 구조의 검증을 위해 (그림 18)과 같은 시험망을 구성하였으며, 서버-1과 서버-2에 OpenVZ를 탑재하고 서버-1에서 서버-2로 온라인 마이그레이션을 수행하는 과정에서 약 3.5초의 다운-타임이 발생하였는데, 이는 동일 LAN 내에서의 마이그레이션과 동일한 결과로써, OpenVZ 특성에 기인하는 것으로 보고 있다. 또한, 모바일 PC의 핸드오프(AP-1→AP-2)의 경우 0.01에서 0.02초 사이의 연결 단절이 발생하였다고 한다[21].

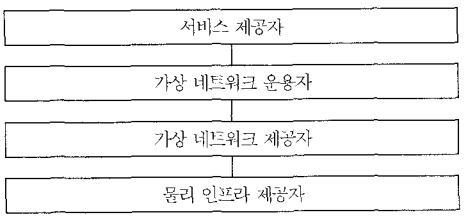
5. 가상 네트워크 아키텍처

현재의 인터넷은 인프라 제공자와 서비스 제공자로 분류할 수 있는데, 이는 현재 부상하고 있는 가상 네트워크 개념을 반영하고 있지 않다. 또한, 클리어링하우스(clearing house), 애그리게이트 매니저(aggregate manager) 등으로 구성되는 GENI의 제어 프레임워크는 순수한 실험용으로 일반 사용자에게 가상화 기반 서비스를 제공하기 위한 가상 네트워크 아키텍처로 적용할 수 없다. 이에 따라, 미래인터넷에서 가상화를 수행하기 위해 인프라 제공자 및 서비스 제공자는 재정의되어야 하고, 새로운 역할 수행자가 필요하다.

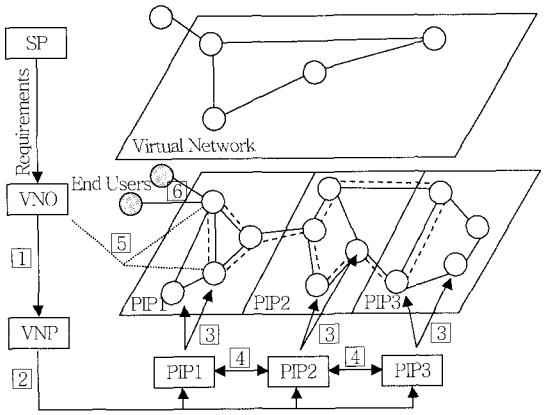
• VNet

이러한 하나의 시도로서, 유럽(독일)에서 (그림 19)와 같이 물리 인프라 제공자(PIP), 가상 네트워크 제공자(VNP), 가상 네트워크 운용자(VNO), 그리고 서비스 제공자(SP)로 이루어지는 4계층의 VNet 구조를 제안하였다. 물리 인프라 제공자는 네트워크 가상화를 지원하는 물리적 인프라(또는 서브스트레이트)를 소유하고 관리하는 역할을 수행하며, 가상 네트워크 제공자는 하나 또는 그 이상의 물리 인프라 제공자의 가상 자원을 모아 가상 네트워크 토폴로지를 제공하며, 가상 네트워크 운용자는 서비스 제공자의 요구에 따라 가상 네트워크 제공자가 제시한 가상 네트워크 위에서 VNet의 설치와 운용을 담당한다. 그리고, 서비스 제공자는 가상 네트워크 위에서 사용자가 원하는 응응 및 네트워크 서비스를 제공한다.

이들 역할 수행자(SP, VNO, VNP, PIP)간 제어 인터페이스의 흐름은 (그림 20)과 같이 동작한다. 즉, SP가 VNO에게 요구사항을 제시하면, VNO는 SP 요구사항에 가상 네트워크에 대한 추가 요구사항 및 제약 사항을 첨부하여 VNP에게 제공한다(interface-1). VNP는 가상 네트워크에 해당하는 하나 또는 그 이상의 PIP에게 필요한 자원을 제시하고(interface-2) PIP 내의 연결을 요청한다(interface-3). 그리고 PIP간 연결도 이루어지면(interface-4) VNO는 가상 네트워크 운용을 시작한다(interface-5). 이후, 최종적으로 사용자는 가상 네트워크 위에서 서비스를 이용한다(interface-6)[22].



(그림 19) VNet 구조



(그림 20) 역할 수행자간 인터페이스

Ⅲ. 결론

지금까지 네트워크 가상화 기술을 주제로 하는 논문이나 자료를 충분히 다룬 것이라고 말하기는 어렵지만, 미래인터넷 관점에서 주요 자료에 대한 분석을 기초로 네트워크 가상화 기술을 호스트 가상화, 링크 가상화, 라우팅 가상화, 가상머신 마이그레이션 그리고 가상 네트워크 아키텍처로 분류하고 각각에 대해 기술 동향을 파악하였다.

여기서, 링크 가상화를 VMware처럼 하이퍼바이저의 I/O 가상화 등의 일부분으로 포함시킬 수 있기 때문에, 이를 호스트 가상화로 묶을 수 있다고 본다. 하지만, 현재 논문 등을 보면 예상외로 많이 다루고 있으며, 자원의 엄격한 분리를 위해 가상 네트워크 인터페이스 역할 또한 중요하기 때문에 하나의 세부 기술로 다루는 것이 적절하다고 생각된다. 따라서, 미래인터넷 관점에서 네트워크 가상화의 세부 기술로 호스트 가상화, 링크 가상화, 라우팅 가상화, 가상머신 마이그레이션 그리고 가상 네트워크 아키텍처를 우선 적용할 수 있는 것으로 본다.

클라우드 컴퓨팅 분야가 현재의 기술에 대한 이슈이라면, 미래인터넷 분야는 미래의 기술에 대한 이슈라고 생각할 것이다. 하지만, 각국에서 진행하고 하고 있는 미래인터넷 테스트베드의 R&D 상황을 보면 네트워크 가상화는 혁신적인 아키텍처 및 서비스를 검증하기 위해 가장 먼저 그리고 지금 바로 해결되어야 할 기술적인 이슈이다. 미래인터넷

테스트베드에서 네트워크 가상화를 시스템 또는 플랫폼 차원에서 지원하지 못한다 하더라도 이미 대부분의 테스트베드에서 부분적으로 지원을 하고 있다. 그리고, 클라우드 컴퓨팅 분야의 핵심 기술인 가상화와 미래인터넷 인프라의 핵심기술이라고 볼 수 있는 네트워크 가상화를 공통 분모로 하여 클라우드 컴퓨팅과 미래인터넷이 랑데뷰하는 그림도 가능할 것이다. 하지만, 그 랑데뷰를 위해 IP 기반의 클라우드 컴퓨팅과 IP 및 non-IP 기반의 미래인터넷의 차이가 어느 정도 영향을 주고 받는지는 사전에 검토가 되어야 할 것이다.

끝으로, 본 고의 후속 작업으로 미래인터넷 테스트베드의 기술 동향을 정리한 후, 이들 작업을 토대로 미래인터넷 테스트베드 플랫폼의 프레임워크 작업을 진행할 예정이다.

● 용 어 해 설 ●

미래인터넷(Future Internet): 현재 인터넷 구조의 한계성을 극복하고 미래의 새로운 요구사항을 수용하기 위해, 기존 인터넷과의 호환성을 필수 조건으로 고려하지 않는 혁신적인 개념(clean-slate)으로 설계될 미래의 새로운 인터넷

네트워크 가상화(Network Virtualization): 공유하는 서브스트레이트(데이터 평면) 위에서 네트워킹 자원의 엄격한 분리 하에 자신의 가상 네트워크를 동적으로 구축하여 네트워크 아키텍처 및 서비스 기술을 전개할 수 있도록 하는 기술

약어 정리

FCoE	Fiber Channel over Ethernet
FIB	Forwarding Information Base
FPGA	Field Programmable Gate Array
GENI	Global Environment for Network Innovation
GNM	Global Node Manager
GPE	General Purpose Processing Engine
GRE	Generic Routing Encapsulation
GRM	Global Resource Manager
HBA	Host Bus Adaptor
IP	Internet Protocol
LNM	Local Node Manager
LRM	Local Resource Manager
MAC	Media Access Control
NIC	Network Interface Card
NPE	Network Processing Engine
OS	Operating System
PCIe	Peripheral Component Interconnect Express
POSIX	Portable Operating System Interface
RIB	Routing Information Base
SOI	Service Oriented Infrastructure
SPP	Supercharging PlanetLab Platform
TCAM	Ternary Content-Addressable Memory
UCS	Unified Computing System
VICTOR	Virtually Clustered Open Router
VMAC-VE	Virtual MAC-Virtual Environment
VMM	Virtual Machine Monitor
VNIC	Virtual Network Interface Card
WP	White Paper

참 고 문 헌

[1] 한국IBM 시스템 테크놀로지 그룹, "가상화 기술의 새로운 패러다임," 한국경제신문, 2007.

[2] 김진미, 안창원, 정영우, 박종근, 고광원, 변일수, 우영춘, "차세대 컴퓨팅을 위한 가상화 기술," 전자통신동향분석, 제23권 제4호, 2008. 8., pp.102-114.

[3] Gernot Heiser, "The Role of Virtualization in Embedded System," IIES08, Apr. 1, 2008.

[4] 박성용(서강대), "Virtualization: Technologies and Trends," ETRI 전문가 초빙 발표 자료, 2008. 4.

[5] http://www.virtualization.info/buyersguide/vmms.asp

[6] VMware white paper, "VMware Infrastructure Architecture Overview," http://www.vmware.com/pdf/vi-architecture-wp.pdp

[7] VMware white paper, "The Architecture of VMware ESXi," http://www.vmware.com/resources/techresources/1009

[8] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, and Tim Harris et al., "Xen and the Art of Virtualization," SOSP03, Oct. 19, 2003.

[9] University of Cambridge, "Overview of Xen 3.0,"

http://www.cl.cam.ac.uk/research/srg/netos/xen/architecture.html
[10] http://wiki.openvz.org/Main_Page
[11] OpenVZ, "OpenVZ User's Guide," 2005.
[12] http://linux-vserver.org/Overview
[13] Sunay Tripathi, Nicolas Droux, and Thirumalai Srinivasan, "Crossbow: From Hardware Virtualized NICs to Virtualized Networks," VISA09, Aug. 17, 2009.
[14] http://www-europe.cisco.com/en/US/prod/collateral/ps10265/ps10276/solution_overview_c22-555987_ps10280_Product_Solution_Overview.html
[15] Sapan Bhatia, Murtaza Motiwala, and Wolfgang Muhlbaur et al., "Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware," Workshop on Real Overlays and Distributed Systems(ROADS), Dec. 2008.
[16] Yong Liao, Dong Yin, and Lixin Gao, "PdP: Parallelizing Data Plane in Virtual Network Substrate," VISA09, Aug. 17, 2009.
[17] Jon Turner, Brandon Heller, and Jing Lu et al., "Supercharging PlanetLab - A High Performance, Multi-Application, Overlay Network Platform," SIGCOMM07, Aug. 27, 2007.
[18] Muhammad Bilal Anwer and Nick Feaster, "Building a Fast, Virtualized Data Plane with Programmable Hardware," VISA09, Aug. 17, 2009.
[19] Matt Kolon, "Intelligent Logical Router Service," Juniper Networks, Oct. 2004.
[20] Christopher Clark, Keir Fraser, Steven Hand, and Jacob Gorm Hanseny et al., "Live Migration of Virtual Machines," NSDI05, May 2, 2005.
[21] Fang Hao, T.V. Lakshman, Sarit Mukherjee, and Haoyu Song, "Enhancing Dynamic Cloud-based Services Using Network Virtualization," VISA09, Aug. 17, 2009.
[22] Gregor Schaffrath, Christoph Werle, and Panagiotis Papadimitriou et al., "Network Virtualization Architecture: Proposal and Initial Prototype," VISA09, Aug. 17, 2009.