ETRI
전자통신동향분석 제25권 제2호 2010년 4월

미래인터넷에서 네트워크 가상화 기술의 적용방안

A Study for Applying for the Network Virtualization Technology in Future Internet

김영화 (Y.H. Kim) 미래네트워크연구부 책임연구원

목 차

서비스 지향 아키텍처를 구현하기 위한 핵심 기술인 가상화 기술 가운데 네트워크 가상화가 미래인터넷에서 중요한 이슈로 부상하고 있다. 이는 다양한 사용자가 공유하는 네트워크 인프라 상에서 혁신적 설계에 바탕을 둔 아키텍처 및 서비스 등을 실현하기 위해 네트워크 가상화가 필수적으로 요구되기 때문이다. 하지만, 네트워크 가상화의 세부적인 기술이 무엇이며, 이들 세부 기술을 어떻게 적용할 것인가에 대해 시스템 개발자는 다양한 기술적 상황을 고려하여 결정해야 한다. 본 논문은 이러한 과정의 일환으로 미래인터넷 테스트베드에서 호스트 가상화, 라우팅 가상화, 슬리버 마이그레이션 등의 네트워크 가상화 세부 기술들에 대한 적용 방안을 다룬다.

Ⅰ. 개요

IT 인프라를 통해 변화하는 비즈니스 우선순위에 따라 재사용 또는 결합 가능한 컴포넌트로 통합하는 표준화된 프레임워크인 서비스 지향 아키텍처를 구현하기 위한 핵심 기술이 가상화이다. 가상화 기술 가운데 네트워크 가상화가 미래인터넷에서 중요한 기술적 이슈로 부상하고 있지만, 아직은 초기 단계이기 때문에 네트워크 가상화의 개념과 세부 기술 등이 모호한 상태이다. 이러한 상황에서 [1]은 미래인터넷 관점에서 네트워크 가상화에 대해 기술 동향을 파악하고, 이에 대한 전반적인 개념과 세부 기술을 다루었다.

[1]에 따르면 네트워크 가상화 기술은 미래인터넷 영역만의 기술이 아니며, 세부 기술을 플랫폼의 특성에 맞게 부분적 또는 제한적으로 다루고 있다는 것이다. 예를 들면, 지금까지 발표된 플랫폼 가운데 두드러지는 것은 SPP[2]와 NetFPGA[3] 기반 플랫폼이다. 네트워크 가상화 관점에서 전자는 링크 가상화가 지원되지 않으며, 라우팅 가상화도 극히 제한적으로 수행되고 있다. 후자는 라우팅 가상화가 초점이기는 하지만, NetFPGA의 하드웨어 그리고 PC의 한계를 벗어나지 못하고 있어, 가상 라우터의 최대 생성 수 그리고 가상 네트워크 인터페이스에 제약이 있다.

그러면, 네트워크 가상화의 세부 기술들을 하나의 플랫폼에서 통합하여 지원할 필요가 있을까 하는 의문이 있을 수 있다. 결론은 통합하여 지원해야 한다는 것인데, 이는 세부 기술들이 서로 밀접하게 연관되어 있기 때문이다. 따라서 플랫폼은 네트워크 가상화 기술의 수평적(특정 계층의 실험) 및 수직적(전체 계층의 실험) 접근을 모두 지원할 수 있어야 하나, 중요한 점은 역방향 호환성, 즉 기존의 IP 트래픽의 수용이 전제가 되어야 한다.

이후, 본 논문의 구성으로 제 Ⅱ장에서는 미래인터넷 테스트베드에서 패킷 구조, 호스트 가상화, 링크 가상화, 라우팅 가상화, 슬리버 마이그레이션, 그리고 가상 네트워크 아키텍처와 같은 다양한 네트워크 가상화의 세부 기술들에 대한 적용방안을 제시한다. 그리고 제 Ⅲ장에서는 이 논문의 결론으로 추후 연구 방향 등을 기술한다.

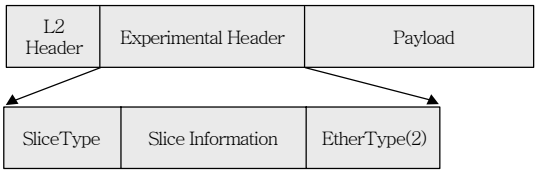
Ⅱ. 네트워크 가상화 기술의 적용

1. IP/non-IP 패킷 구조

대부분의 미래인터넷 테스트베드 상에서 교환되는 패킷은 현재 이더넷 패킷 구조를 그대로 적용하고 있다. 이는 해당 슬라이스 내의 슬리버들이 모두 IP 위에서 동작하기 때문에 이더넷 패킷 구조를 확장할 필요가 없다. 하지만, 미래인터넷에서는 무선 및 이동의 자체 확장 및 유연성을 지원하기 위해 All-IP를 주장하지 않는다. 따라서, non-IP 지원도 테스트베드의 중요한 요소이며, [4]와 같은 극소수의 플랫폼에서는 이를 지원할 수 있다고 주장하고 있다.

또한, GENI의 설계 문서에서는 이더넷 패킷의 페이로드 앞에 슬리버 식별자, 가상 인터페이스 식별자, 발신 위치 식별자, 수신 위치 식별자 등을 포함하는 이더넷 확장 구조를 다루고 있지만[5], 아직은 적용되지 못하고 있다. 또한, 미래인터넷 테스트베드에서는 가상 MAC 기능이 필수라고 볼 수 있는데, 슬리버 식별자를 가상 MAC 주소로 대응시키는 현재의 구조를 그대로 사용하는 임시 방편도 있다. 하지만, 이는 가상 MAC을 슬리버로 보기 때문에 슬리버 개념을 혼돈하게 만들고 하나의 가상 MAC에서 다수의 슬리버를 갖는 구조를 지원할 수 없다.

본 논문에서 좀더 다양한 상황을 지원할 수 있는 패킷 구조를 제안한다. 그 배경에는 본 논문의 뒷부분에서 언급한 가상 네트워크 아키텍처와 관련이 있다. 즉, 가상 인터페이스, 슬리버, 가상 노드, 가상 네트워크, 슬라이스, 가상 네트워크 제공자 등으로 구성되는 가상 네트워크 아키텍처를 반영하기 위함이다. 또한, 사용자가 자신의 슬라이스에서 다양한 제어 정보를 구성할 수 있는 확장성을 지원하는 것이 필요하다. 예를 들면, 어떤 슬라이스는 슬라이스 식



(그림 1) 확장 이더넷 패킷 구조

별자만이 필요할 수 있고, 다른 슬라이스는 슬리버 식별자, 가상 인터페이스 식별자, 발신 위치 식별자, 수신 위치 식별자 정보 등이 요구될 수 있다. 이러한 요구사항을 반영한 패킷 구조는 (그림 1)과 같다.

(그림 1)의 구조에서 이더넷 헤더의 EtherType은 미래인터넷 트래픽용 유형을 표시한다. 실험 헤더의 슬라이스 유형은 다양한 슬라이스 제어 정보의 유형을 표시하고, 슬라이스 정보는 해당 슬라이스 유형에 대한 구체적인 제어 정보를 나타낸다. 그리고 실험 헤더의 EtherType은 실제 페이로드 유형을 표시한다. 그 값은 현재의 EtherType에서 정의된 값이 될 수 있으며, 해당 슬라이스의 성격에 따라 새로이 할당될 수 있다.

2. 호스트 가상화

가. 호스트 가상화 유형

주요 요구사항들에 대한 전가상화, 반가상화 그리고 OS 기반 가상화로 분류한 호스트 가상화 유형간 비교는 <표 1>과 같다.

<표 1>에서, 슬리버 OS 수정을 제외하고 자원 가상화, 슬리버 마이그레이션 요구사항들은 호스트 가상화 유형간의 차이가 크게 나타나지 않는다. 다만, OS 기반 가상화에서 자원 가상화와 슬리버 마이그레이션 요구사항들은 OpenVZ에서만 가능하다는 점이다. 슬리버 생성 수 요구사항은 반가상화 방식의 경우 특별히 명시하기 어렵지만, 메모리 등 하드웨어 자원들이 슬리버 생성 수만큼 지원할 수 있으면 크게 문제되지 않을 것이다. 슬리버 OS 요구사항은 필요한 슬리버 OS를 정확히 제시하기 어렵지만, 현재로서는 Linux로 충분할 것으로 판단된다. 성능 요구사항의 경우, 전가상화의 하드웨어 에뮬레이션으로 인한 오버헤드 또는 Xen의 SMP 지원 여부 등 공개 자료가 명확하지 않은 상태에서 그리고 미래인터넷 테스트베드의 특성(성능 보다는 유연성)에서 호스트 가상화 방식을 결정하는 데 성능이 결정 요소로 작용하기 어렵다. 예를 들면, [6]은 전가상화의 대표격인 VMware와 반가상화의 대표격인 Xen의 성능 비교에서 오히려 VMware가 컴파일링, CPU, 대역폭, 가상머신 등 다양한 척도에서 Xen 보다 우수하다고 주장한다. 참고로, Xen의 최근 버전에서는 QEMU 기능을 활용하여 전가상화 방식도 지원하고 있다.

하지만, 타 슬리버 영향 유무 요구사항부터는 상황이 다르다. OS 기반 가상화의 가장 큰 약점이 시스템 OS의 장애로 슬리버에 영향을 줄 수 있다는 점이다. 이는 반가상화의 경우, Dom0 OS의 영향을 배제할 수 없지만, 이는 큰 문제가 되지 않을 것으로 판

〈표 1〉 호스트 가상화 유형간의 비교

요구사항	전가상화	반가상화	OS 기반 가상화
슬리버 OS 수정	수정할 필요 없음	수정해야 함	수정할 필요 없음
자원 가상화	CPU, 메모리, I/O	CPU, 메모리, I/O	CPU, 메모리, I/O
슬리버 마이그레이션	지원	지원	지원(OpenVZ)
슬리버 생성 수	10개 이상	? (메모리 등에 의존)	제약 없음
슬리버 OS	Linux, Windows, Solaris, Netware	Linux, NetBSD, FreeBSD, Solaris	Linux
성능	H/W 에뮬레이션으로 성능 저하	전가상화보다 우위	제약 없음
타 슬리버 영향 유무	영향 없음	?	시스템 OS 영향 가능
프로세서	Intel x86, AMD x64	Intel x86, IA64, ARM	Intel x86

단된다. 프로세서 요구사항은 x86 뿐만 아니라 해당 네트워크 프로세서(예: 옥테온 NP)도 지원되어야 하나, 현재 어떠한 호스트 가상화 유형도 NP를 지원하지 않기 때문에 어떠한 경우든 NP를 지원하기 위한 추가 작업이 요구된다. 이러한 분석을 토대로 미래 인터넷 테스트베드에서 호스트 가상화 유형은 성능 문제가 이슈가 되지 않는 한, OS 기반 가상화 보다는 반가상화 방식이 더 적합하며, 반가상화 보다는 전가상화 방식이 더 적합하다. 하지만, 성능 문제가 심각한 이슈로 등장할 때에는 전가상화 보다는 반가상화 방식이 더 적합할 것이다.

나. 호스트 가상화 구현 방식

호스트 가상화의 실제 구현 방안으로, 자체 구현 방식, 공개 소스 방식(예: Xen), 그리고 상용 소스 방식(예: VMware)이 있을 수 있다. 미래인터넷 테스트베드의 결과물로써 파급 효과를 고려한다면 당연히 자체 구현 방식이 요구된다. 하지만, 다양한 하드웨어 자원의 가상화 기능과 타이머, IPC, 마이그레이션 그리고 전체적인 제어 등 가상화 지원 기능과 개발 인력, 기간 그리고 안정화 등을 고려할 경우, 자체 구현 방식은 단기간으로는 상당한 무리가 따른다. 공개 소스 방식은 호환성 측면에서는 다른 어떤 방식과 비교하더라도 상대적 우위에 있으나 연구 결과물을 라이선스 규약에 따른 공개의 의무가 있다. 예를 들면, GPL/LGPL 기반 소스의 경우 만일 리눅스 커널의 수정이 있을 때는 수정한 커널 소스를 공개해야 한다. 하지만, 응용 프로그램, 독자 개발 라이브러리, 동적 링크 디바이스 드라이버 등의 소스는 공개할 의무가 없으며, 이들에 대한 IPR 획득에도 문제가 없을 것으로 판단된다[7].

이에 반해 상용 소스 방식은 영향력과 소요 비용을 제외하면 개발 인력, 개발 기간, 안정화에 우위가 있다고 볼 수 있다. 즉, 테스트베드가 단기 및 중기의 결과물이 요구되고 실험적 측면이 강하면 공개 소스 방식이 타당하며, 단기 및 중기의 결과물과 제품적 특성이 요구되면 상용 소스 방식이 타당하고, 그리고 장기적인 결과물을 요구할 경우 자체 구현 방식이 타당할 것이다. 따라서, 미래인터넷 테스트베드는 실험적 특성이 강하지만 제품적 특성을 포함하는 것은 현실적으로 무리가 있으며, 혁신적인 아키텍처 및 서비스의 검증을 위해 가능한 한 조기에 테스트베드 개발 작업이 마무리되어야 하고 테스트베드의 저비용 측면에서 공개 소스 방식이 합당하다. 또한, 공개 소스의 재활용을 통해 일차적으로 구현해야 하는 다른 세부 기술에 초점을 맞출 수 있으며, 네트워크 가상화 기술의 이해도를 증가시킬 수 있는 장점이 있다.

다. 적용방안

결론적으로 호스트 가상화 유형 및 구현 방식 등을 전체적으로 놓고 볼 때, 호스트 가상화는 일차적으로 x86(프로세서 카드용) 및 NP 모두 Xen을 사용·확장하는 것이 단기적인 해결책으로 적절한 답이라고 할 수 있겠다. 하나의 예로, 라우터 가상화를 위해 멀티코어 당 하나의 가상 라우터를 구현한다고 가정해 보자. 프로세서 카드에서 제어 및 관리 평면의 가상화 그리고 라인카드에서 SP 및 FP 가상화를 위해 Xen을 활용하는 것이다. 이때, 라인 카드에서 링크 가상화 및 라우터 가상화를 위한 제약 및 한계 등이 확인될 수 있을 것이다. Xen을 사용하여 테스트베드를 구축하면서 노하우 및 제반 기술을 습득한 이후, 필요한 시점에서 자체 구현 방식으로 개발해 나가는 것이 적절하다.

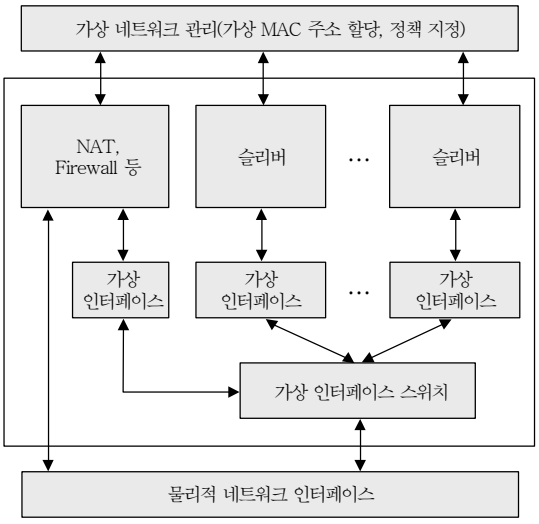
3. 링크 가상화

현재와 같은 물리적 포트 단위의 MAC 인터페이스 설정은 대역폭 운용의 비효율성이나 응용별 동적 I/O 구성을 불가능하게 한다. 하지만, 미래인터넷에서 슬리버 단위의 자원 할당을 제어하기 위해 하나의 물리적 네트워크 인터페이스의 최대 대역폭 범위 내에서 가상 MAC 주소 할당을 통해 다수의 I/O 접속을 동적으로 제어하는 가상 네트워크 인터페이스 기술이 필수적으로 요청된다. 따라서, 미래인터넷 테스트베드에서 MAC 주소 학습은 하나의 물리적

노드에서 물리적 및 가상적 MAC 주소 학습 두 가지 모두 지원되어야 한다. 이를 위해, 물리적 네트워크 인터페이스는 이전의 동작에 추가하여 가상 인터페이스를 통해 특정 슬리버로 패킷을 전송할 수 있어야 하며, 반대로 해당 슬리버는 가상 인터페이스를 통해 물리적 네트워크 인터페이스 상에서 패킷을 그대로 전송할 수 있어야 한다.

또한, 노드에 슬리버를 설치할 때 슬리버를 관리하는 서버가 가상 MAC 주소를 할당하고 각 가상 네트워크 인터페이스 단위로 정책을 설정할 수 있어야 하며, [8]과 같이 NAT, firewall 그리고 VLAN을 통해 물리적 노드 내부 또는 네트워크 차원에서 가상 와이어를 구축할 수 있어야 한다.

하지만, 모든 슬리버가 엄격한 자원 분리를 요구하지 않을 수 있기 때문에 링크 가상화 지정은 사용자의 결정사항으로 할 수 있어야 한다. 예를 들면, 사용자는 기능 검증만을 수행하는 실험을 위해 서브스트레이트에서 패킷의 교환 과정만을 보장할 수 있다면 반드시 링크 가상화를 적용할 필요가 없을 수 있다. 또한, 하나의 슬리버에서 다수의 가상 MAC 주소 적용도 선택할 수 있어야 하며, 반대로 다수의 슬리버가 하나의 가상 MAC 주소의 활용도 제약이 없어야 할 것이다. (그림 2)는 이러한 링크 가상화에 대한 개념적 구조를 나타낸다.



(그림 2) 링크 가상화의 개념적 구조

4. 라우팅 가상화

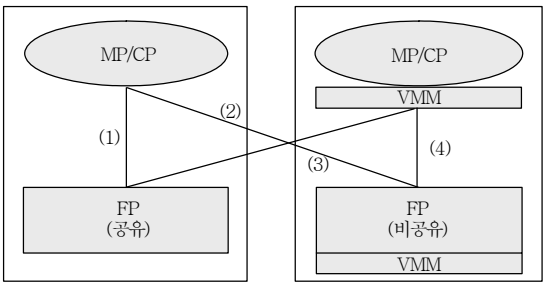
미래인터넷 테스트베드의 라우팅 가상화는 하나의 물리적인 라우터에서 자원을 엄격히 분리하여 다수의 가상 라우터를 구성하는 기술이다. 여기서, 슬리버 당 자원의 분리 이슈만을 제외하면 Juniper에서 현재 지원하고 있는 "논리적 라우터"와 개념적으로 유사하다고 볼 수 있다. 즉, 하나의 물리적 라우터에서 다수의 논리적 라우터를 구성하기 위해 제어평면, RIB 및 FIB를 가상 라우터 단위로 분리한다[9]. 하지만, 미래인터넷 테스트베드에서 라우팅 가상화는 T1600, JCS1200 등의 Juniper 장비[10] 이상으로 엄격한 자원 분리와 다양한 네트워크 가상화의 세부 기술에 기반한 가상 네트워크를 구성할 수 있어야 한다.

이러한 조건 하에서 NP 기반 노드에서 간단한 가상 라우터의 구현 방법은 다음과 같다. 즉, 하나의 멀티코어에 FP(가상 라우팅 패킷 처리), CP(가상 라우팅 프로토콜), MP(가상 라우팅 에이전트)를 수용하는 것이다. 따라서, 하나의 NP에서 또는 하나의 라인카드에서 라우팅 가상화를 위한 기본적 오버헤드를 제외하면, NP당 약 15개의 가상 라우터를 구성할 수 있게 된다. 일차적인 가상 라우터의 구현 후, 좀더 확장하여 하나의 멀티코어에서 다수의 가상 라우터를 구성하거나 라인카드 또는 프로세서 카드간 FP, CP, MP를 분리하여 다수의 가상 라우터를 구성할 수 있다. 즉, 라인카드는 FP 가상화 관련 슬리버를 처리하고, 프로세서 카드는 CP 및 MP 가상화 관련 슬리버를 처리하도록 하는 것이다. 여기서 주의할 점은 기본적으로 모든 방법에 있어서 가상 라우터 당 하나 또는 다수의 가상 네트워크 인터페이스 기능을 지원한다는 것을 전제조건으로 한다.

다음은 FP 가상화 방법으로, FP라 함은 고속의 패킷 처리를 하는 부분을 말한다. 예를 들면, FP 가상화를 지원하지 않는 SPP의 경우, 수신 모듈, 송신 모듈, 파싱 모듈, 룩업 모듈 등의 다양한 패킷 처리 모듈들을 파이프라인 형태로 구현하여 프로세서 카드 상의 다양한 슬리버가 라인카드 상의 FP 기능을 공유한다. 따라서, FP 가상화라고 함은 슬리버 단위

로 이들 패킷 처리 모듈들의 독점적 실행을 가능하게 하는 것이다. 또 다른 예로 클릭(click)의 IP 라우터 구성시 사용하는 패킷 포워딩 모듈들(Classifier, CheckIPHeader, LookupIPRoute, FixIPSrc, DecIPTTL 등)을 슬리버 단위로 수행시키는 것이다. 이러한 예로 Linux 커널에서 슬리버 단위로 클릭 모듈들을 구성하여 FP를 가상화 하였다[11]. 하지만, NP에서 FP 가상화는 NP의 고유 목적인 패킷 처리의 고속화의 바탕 위에서 FP의 유연성을 제공할 수 있는 방안이 될 것이다.

그리고, 하나의 미래인터넷 인프라 장비는 역방향 호환성, 즉 기존의 IP 트래픽의 수용을 전제로 해야 하기 때문에 (그림 3)과 같이 4개의 패스를 통해 MP, CP 및 FP의 단위 기능들이 독립적으로 동작할 수 있는 비-슬리버 기반 라우터와 슬리버 기반 가상 라우터를 동시에 지원하는 구조가 되어야 할 것이다.



(a) 비-슬리버 기반 라우터 (b) 슬리버 기반 가상 라우터

(그림 3) 미래인터넷 인프라 장비의 개념적 구조

5. 슬리버 마이그레이션

클라우드 컴퓨팅 분야에서 VM 마이그레이션은 서비스 연속성을 유지하기 위한 주요 세부 요구사항 중의 하나이다. 이는 클라우드 컴퓨팅에서 VM은 네트워크 내부의 자원이 아니라 해당 서비스를 실행하는 서버의 자원으로, VM의 다운은 수많은 서비스 이용자의 서비스 접근을 불가능하게 하여 사회·경제적으로 문제를 야기할 수 있기 때문이다.

미래인터넷의 테스트베드에서 슬리버 마이그레이션 상황은 다르다. 기본적으로 슬리버의 등록은 사용자와 클리어링 하우스(clearing house) 사이에서 이루어지며, 하나의 미래인터넷 서브넷에서 동일한 슬리버는 다수의 노드에서 존재할 것이다. 그리고 특정 슬리버 다운은 해당 응용의 서비스 지원 불가를 의미하지 않는다. 그리고 데이터 평면을 통한 슬리버의 마이그레이션 대신 슬리버의 재설치로 해결될 수 있을 것이다. 하지만, 해당 슬라이스에서 특정 슬리버의 다운으로 서비스 지속 상황에 심각한 영향을 줄 수 있을 때(예: 가상 NAT 및 DNS 운용), 또는 하나의 슬라이스를 구성하는 슬리버의 개수가 임계치 이하로 떨어지는 경우에 슬리버 마이그레이션을 수행할 수 있다. 또한, 부하 분산을 위해 기존의 슬라이스에 새로운 슬리버를 추가하는 것도 넓은 의미로 슬리버 마이그레이션으로 해석할 수 있다. 이를 위해, 마이그레이션 정책, 터널링 및 포워딩, 슬리버 전달 네트워크, 그리고 마이그레이션 절차로 이루어지는 네트워크 가상화 프레임워크는 다음과 같이 적용할 수 있다.

가. 마이그레이션 정책

슬리버를 한 위치에서 다른 위치로 이전하기 전에 적절한 사전 조치가 수행되어야 한다. 이를 테면, 클리어링 하우스, 애그리게이트 매니저(aggregate manager) 또는 기타 서버에서 해당 슬리버의 이동으로 야기되는 자원 할당이나 이전 노드 선정(사용자 위치 기반 노드 또는 백업 노드 등) 그리고 부하 분산 방식 등을 결정하는 것이다. 이러한 오버헤드는 슬리버의 특성에 따라 정적, 동적 그리고 하이브리드 방식 등을 적용할 수 있을 것이다. 예를 들면, 신속한 후속 패킷 처리가 필요한 슬리버는 사전에 오버헤드를 미리 수행할 수 있으며, 슬리버의 마이그레이션이 필요한 상황이 감지된 경우에 오버헤드를 수행하거나, 정적 및 동적을 결합하여 세부 오버헤드의 성격에 따라 처리 작업을 분리하여 진행할 수 있다. 하지만, 미래인터넷 테스트베드에서 일차적으로 초기에는 정적 방식을 적용 대상으로 하는 것이 바람직하다. 즉, 슬리버의 성격에 따라 사전에 마이그레이션 여부와 이전할 노드의 리스트를 결정하고, 이후 필요하면 동적 방식으로 확장한다.

나. 터널링 및 포워딩

슬리버 마이그레이션을 위해 새로이 구성된 슬리버와 사용자간 그리고 슬리버간 패킷 교환을 위해 이전 터널링과 포워딩을 삭제하고 새로운 터널링 및 포워딩을 설정해야 한다. 이러한 터널링 및 포워딩을 슬리버 내부에서 해결하거나 또는 슬리버 외부의 기능을 활용하여 적용할 수 있다. 슬리버 외부의 기능을 활용할 경우 [12]의 OpenFlow 기능을 일차적으로 적용하며, 이 논문에서는 슬리버 내부에서 터널링 및 포워딩 수단을 제공하는 것은 논외로 한다. 다만, 사용자가 원하는 방법을 지정할 수 있어야 한다. OpenFlow 기능을 적용할 때, 새로이 구성되어야 하는 가상 네트워크 토폴로지 뷰와 연계하여 패킷 포워딩을 설정하는 것이 효과적이다. 따라서, 노드 선정, 가상 네트워크의 토폴로지 구성, 그리고 OpenFlow 기반 터널링 및 포워딩 제어 과정을 연속 동작으로 수행할 수 있어야 한다.

다. 슬리버 전달 네트워크

슬리버의 전달은 슬리버 저장소의 간접 전달과 해당 노드의 직접 전달의 두 가지 방식이 있다. 슬리버 저장소의 간접 전달은 슬리버의 최초 구성과 유사하다. 즉, 마이그레이션이 발생하기 전의 슬리버를 구성할 때 슬리버 저장소에서 그 노드로 다운로딩하는 것과 동일하게 처리하는 것이다. 이는 별도의 슬리버 전달 네트워크는 필요하지 않지만, 이전 슬리버의 상태 정보(사용자 및 슬리버 상태 정보 등)가 필요 없는 부하 분산을 위해 적용할 수 있다. 또한, 이전 슬리버의 상태 정보를 유지할 수 있는 경우에 직접 전달의 보조 수단으로도 사용할 수 있다. 해당 노드의 직접 전달은 해당 슬리버에 대한 전체 이미지(상태 정보 포함)를 마이그레이션이 발생하기 전 노드에서 새로운 노드로 이전하는 것이다. 이를 위해 이들 노드간 마이그레이션 터널을 설정하고, 이후 절차는 VM 메모리 복사, 이전 VM 중단, 마이그레이션 완료 확인, 그리고 신규 VM 활성화 등으로 이루어지는 [13]의 Xen 마이그레이션 절차와 유사하게 수행하도록 하는 것이다. 이후, 이들 노드간 마이그레이션 터널을 삭제한다. 하지만, 노드간 직접 전달이 불가능한 상황을 대비하기 위해 마이그레이션이 발생하기 전 노드는 지속적으로 슬리버 저장소에 상태 정보를 반영하고, 마이그레이션 트리거링을 해당 노드와 슬리버 저장소가 협력하여 결정한다.

라. 마이그레이션 절차

지금까지 언급된 내용을 기초로 미래인터넷 테스트베드에서 슬리버 마이그레이션은 서버에서 전체적으로 제어·관리하도록 하며, 그 절차는 다음과 같은 순서로 진행한다.

① 마이그레이션 정책 결정
② 노드 선정 및 자원 예비 할당
③ 마이그레이션 신호 감지 및 슬리버 전달 방식 결정
④ 사용 자원 최종 확인
⑤ 슬리버 전달 네트워크 설정
⑥ 슬리버 이미지 복사
⑦ 이전 슬리버 중단
⑧ 슬리버 마이그레이션 완료 확인
⑨ 슬리버 전달 네트워크 해제
⑩ 신규 슬리버 활성화
⑪ 링크 가상화 활성화
⑫ OpenFlow 기반 포워딩 재조정

위 절차 중, 마이그레이션 신호의 감지 및 슬리버 전달 방식의 결정은 해당 노드와 서버간에 특정 프로토콜을 통해 해결한다. 즉, 일차적으로 해당 노드에서 마이그레이션 신호를 감지하고 직접 전달을 통해 마이그레이션을 수행하고, 이 과정에서 문제가 발생할 경우 슬리버 저장소를 통한 간접 전달 방식을 적용한다. 여기서 새로운 노드로 마이그레이션된 슬리버의 주소 할당은 상황에 따라 다를 수 있다. 예를 들면, 동일한 서브넷에서의 마이그레이션은 주소를 변경할 필요가 없으나, 그 이외에는 주소를 변경해야 한다. 그리고 사용하는 주소가 사설 주소인 경우에는 추가적으로 공공 주소가 할당되고, 이 공

공 주소는 필요하면 관련 노드들에게 통보되어야 한다. 이를 위해, 테스트베드의 게이트웨이 노드에 NAT 및 기존 라우팅 프로토콜 등이 탑재되어야 한다.

6. 가상 네트워크 아키텍처

이동망에서 인프라 제공자인 이동통신망사업자(MNO)의 설비(무선 접속 등)를 활용하여 사용자에게 이동통신 서비스를 제공하는 가상이동망사업자(MVNO)가 있다. 이 사업자는 이동통신 서비스를 제공하기 위해 필수적인 주파수를 보유하고 있지 않지만, 주파수를 보유하고 있는 이동통신망사업자의 망의 무선 및 유선 설비를 활용하여 독자적인 이동통신 서비스를 제공하는 사업자를 의미한다. 이러한 MVNO의 도입으로 고객의 선택권의 확대, 서비스 종류의 다양화, 요금인하 효과 등이 있는데, MVNO는 하드웨어 장비를 보유하고 있는 인프라 제공자의 설비를 이용하여 사용자에게 가상 네트워크 서비스를 제공하는 가상 네트워크 제공자와 유사하다고 볼 수 있다[14].

현재의 인터넷은 인프라 제공자와 서비스 제공자로 분류할 수 있는데, 이는 현재 미래인터넷에서 부상하고 있는 가상 네트워크 개념을 반영하고 있지 않다. 또한, 클리어링 하우스, 애그리게이트 매니저 등으로 구성되는 GENI의 제어 프레임워크는 순수한 실험용으로 일반 사용자에게 가상화 기반 서비스를 제공하기 위한 가상 네트워크 아키텍처로 적용할 수 없다[15]. 이에 따라 미래인터넷의 조기 시장 대응 및 시장 개척을 위해 유선망 및 무선망을 아우르는 가상 네트워크 아키텍처를 (그림 4)와 같이 제안한다.

(그림 4)에서 "계층적 가상 네트워크 제공자"는 하나의 가상 네트워크 제공자가 하나 또는 그 이상의 물리적 네트워크 제공자의 설비를 활용하여 사용자 및 서비스 제공자에게 가상 네트워크 서비스를 제공한다. 그리고 하나 또는 그 이상의 가상 네트워크 제공자 위에 차상위의 가상 네트워크 제공자가 존재하여 차하위의 가상 네트워크 제공자가 공통으로 원하는 기능을 지원하며, 차상위에 차차상위의

(그림 4) 가상 네트워크 아키텍처의 개념 구조

가상 네트워크 제공자가 중첩으로 구성될 수 있다. 또한, 역방향의 호환성을 고려하여 사용자, 서비스 제공자, 그리고 바로 물리적 네트워크 제공자의 구조를 수용할 수 있어야 한다. 그리고 특정 사용자(예: 실험자)는 서비스 제공자의 중개 없이 바로 가상 네트워크 제공자의 서비스를 제공 받을 수 있어야 한다.

Ⅲ. 결론

미래인터넷에서 호스트 가상화로 시작된 네트워크 가상화 기술은 아직은 초기 상태이다. 어떤 점에서는 호스트 가상화를 포함하여 링크 가상화, 라우팅 가상화, 그리고 가상머신(또는 슬리버) 마이그레이션들로 이루어지는 네트워크 가상화가 새로운 기술이라기 보다는 최근 기술들의 통합 성격이 짙다고 볼 수 있다. 하지만, 네트워크 가상화는 미래인터넷의 새로운 패러다임의 창을 열 수 있는 중요한 도구로 등장할 가능성이 높은 기술이다. 이러한 시도로써, 유럽의 4WARD, 미국의 FIND 및 GENI에서 가상화 관련 아키텍처 및 테스트베드의 R&D에 주력하고 있다.

이제 국내에서도 미래인터넷 관련 프로젝트를 본격적으로 진행하기 위한 출발점에 서 있다. 미래인터넷 테스트베드를 개발하는 과정에서 주요 사용자인 국내 학계에서 유선 및 무선 분야에서 일차적으로 시도하였고 익히 잘 알려진 저비용의 PlanetLab 제어 프레임워크의 활용·확장을 통해 단계적인 네트워크 가상화 기술을 개발하여 궁극적으로 미래인

터넷 인프라 장비로 발전해 나가는 방향을 권고하고 싶다. PlanetLab 제어 프레임워크 하에 우선 NetFPGA 기반 PC 노드를 구성·공개하고, 이후에 ATCA 기반 노드와 무선·이동부분을 추가로 구성·공개하는 것이다. 이는 무엇보다도 테스트베드가 일차로 작업이 되어 혁신적인 아키텍처 및 서비스를 연구하는 사람들이 우선적으로 사용할 수 있어야 하며, 이를 통해 이전 단계의 미비한 점을 보완하고 새로운 기능의 추가를 반복하면서 미래인터넷 테스트베드의 실사용자 층을 점차적으로 확대해 나가는 것이 더욱 중요한 과정이기 때문이다. 이를 위해서는 출연기관 및 학계가 함께 고민하고 문제를 해결하고자 하는 적극적인 의지와 노력이 필요하다.

OpenFlow 및 FlowVisor와 같은 미래인터넷의 상용장비 접목기술을 현재의 인터넷 시스템(예: 패킷-광 통합 스위치)에 적용하는 것도 미래인터넷 인프라 기술을 조기에 구축할 수 있는 또 하나의 방법이다. 중장기적으로는 다양한 네트워크 가상화의 세부 기술들을 하나로 엮은 통합 노드를 구성하고, 상위에서 이들 노드들을 가상 네트워크 아키텍처로 묶는 큰 그림을 제안한다. 또한, 본 논문에서는 무선을 고려하지 않았지만, [17]에서 미래인터넷 무선 단말을 위하여 호스트 가상화로 OS 기반 가상화 유형(VMware workstation)과 MS Windows 시스템 OS를 사용하였다. 즉, 무선 분야를 반영한 통합 미래인터넷 테스트베드에 대한 전체적인 프레임워크 확인 작업도 서둘러야 할 것이다.

● 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

ATCA	Advanced Telecommunication Computing Architecture
CP	Control Plane
CPU	Central Process Unit
DNS	Domain Name Service
FIB	Forwarding Information Base
FIND	Future Internet Design
FP	Forwarding Plane
GEN	Global Environment for Network Innovation
IP	Internet Protocol
IPC	Inter-Process Communication
IPR	Intellectual Property Rights
(L)GPL	(Less) General Public License
MAC	Media Access Control
MP	Management Plane
MVNO	Mobile Virtual Network Operator
NAT	Network Address Translation
NetFPGA	Network Field Programmable Gate Array
NP	Network Processor
OS	Operating System
RIB	Routing Information Base
SPP	Supercharging PlanetLab Platform
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Monitor

참 고 문 헌

[1] 김영화, "미래인터넷의 네트워크 가상화 기술동향," ETRI, 전자통신동향분석, 제25권 제 1호, 2010. 2., pp.132-147.

[2] Jon Turner, Brandon Heller, and Jing Lu et al., "Supercharging PlanetLab – A High Performance, Multi-Application, Overlay Network Platform," SIGCOMM07, Aug. 27, 2007.

[3] Muhammad Bilal Anwer and Nick Feaster, "Building a Fast, Virtualized Data Plane with Programmable Hardware," VISA09, Aug. 17, 2009.

[4] 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.
[5] Jennifer Rexford and T.V. Lakshman et al., "Life of a Packet within a GENI Experiment," GENI, GDD-07-47, Apr. 2, 2007.
[6] VMware white paper, "A Performance Comparison of Hypervisors," http://www.vmware.com/pdf/hypervisor_performance.pdf
[7] 정보통신부, "오픈소스 SW 라이선스 가이드," 2007. 11.
[8] Sunay Tripathi, Nicolas Droux, and Thirumalai Srinivasan, "Crossbow: From Hardware Virtualized NICs to Virtualized Networks," VISA09, Aug. 17, 2009.
[9] Matt Kolon, "Intelligent Logical Router Service," Juniper Networks, Oct. 2004.
[10] 김병장, "HSN을 위한 메트릭스 기술," HSN2010, Feb. 2010.
[11] Eric Keller and Even Green, "Virtualizing the Data Plane Through Source Code Merging," PRESTO08, Aug. 22, 2008.
[12] http://www.openflowswitch.org/, "OpenFlow Switch Specification," Version 1.0.0, Dec. 31, 2009.
[13] Christopher Clark, Keir Fraser, Steven Hand, and Jacob Gorm Hanseny et al., "Live Migration of Virtual Machines," NSDI05, May 2, 2005.
[14] 디지털타임스, "MVNO(가상이동통신망사업자)," 2007. 7. 9.
[15] GPO, "GENI Control Framework Architecture," GENI-ARCH-CP-0.14, Oct. 20, 2008.