요약

다양한 산업 분야에서 요구되는 이질적인 요구사항은 차세대 모바일 네트워크(5G 및 6G)를 넘어 네트워킹 패러다임의 변화를 주도하여 생산성, 성능 및 효율성을 향상시키는 데 중요한 운영 경쟁력을 제공했습니다. 또한, 글로벌 디지털 혁명(예: 인더스트리 4.0)과 연결된 세계에서 네트워크 가상화, 높은 신뢰성과 고성능 통신이 이동통신 사업자에게 중요한 요소가 되었습니다. 핵심 서비스를 악화시킬 수 있는 부정적인 영향을 최소화하기 위해, 서비스 수준 협약(SLA) 및 핵심 성과 지표(KPI)를 충족시키기 위한 핵심 기술로 네트워크 슬라이싱[1]이 널리 인정받고 있습니다. 이런 맥락에서, 프로그래밍이 가능한 데이터 플레인[2]을 도입하여 유연한 서비스 품질(QoS) 약속을 시행하고, 고성능 패킷 처리 및 실시간 모니터링 기능을 제공하는 것이 필수적입니다. 이를 위해 이 논문은 eXpress Data Path(XDP), extended Berkeley Packet Filter(eBPF), 스마트 네트워크 인터페이스 카드(SmartNICs)를 포함한 하드웨어 기반 기술 세트를 활용하여 네트워크 기능을 오프로드[3]하는 혁신적인 프레임워크를 설계, 프로토타이핑, 평가하는 데 중점을 두고 있습니다. 이는 고성능 pre-6G 프론트, 미드 및 백홀 [4] 네트워크 통신을 제공하고 Linux 커널로 인한 오버헤드[5]를 줄이는 것을 목표로 합니다. 제안된 솔루션은 Linux 커널을 우회하고 통신을 가속화하며 네트워크 슬라이스 제어 및 실시간 모니터링 기능을 제공하는 기반으로 구현되었습니다. 이 프레임워크의 주요 목적은 향후 6G 인프라에서 6G KPI를 보장하고 시스템 과부하를 방지하여 네트워크 통신을 보장하는 것입니다. 인더스트리 4.0 서비스에 대한 솔루션의 실증적 검증은 25Gbps, 초당 2천만 패킷, 0% 패킷 손실, 0.1ms의 지연 시간, CPU 부하 10% 미만과 같은 패킷 처리 측면에서 주요 성능 개선을 보여줍니다.

Xdp, eBPF에 대하여

- https://pak-j.tistory.com/62 - xdp
- https://velog.io/@hellonewtry/eBPF-살펴보기 - eBPF

서론

---

[1] 물리적 네트워크 인프라를 여러 가상 네트워크로 분할하는 기술, 각 가상 네트워크는 독립적으로 운영, 네트워크 리소스의 효율적인 관리, 유연성과 효율성 향상

[2] 패킷을 빠르게 처리하고 전달하는 네트워크 부분

[3] 오프프로세서로부터 전용 하드웨어나 다른 장치로 이전하여 메인 프로세서의 부담을 줄이는 것

[4] 6G 이전의 이동통신 네트워크 구조이며 프론트홀(무선 기지국과 중앙처리 장치 사이의 데이터 전송), 미드 홀(네트워크 중간 단계로 기지국과 코어 네트워크 사이의 데이터 전송), 백홀(코어 네트워크와 인터넷, 데이터 센터 등의 인프라 사이의 데이터 전송)이 있다

[5] 어떤 작업을 수행할 때 부수적으로 발생하는 추가적인 자원 사용

5세대(B5G)와 6세대(6G) 이동통신 네트워크는 4G 및 5G와 같은 현재의 이동통신 기술을 뛰어넘는 고급 기능을 지원하여 매우 까다로운 응용 프로그램을 수용할 것으로 예상됩니다. 이러한 응용 프로그램이 차세대 네트워크(NGN)에 부과할 상당한 도전 과제와 다양한 요구 사항을 감안할 때, 미래의 이동통신 네트워크 인프라는 고속으로 네트워크 패킷을 처리하고 유연한 방식으로 서비스 품질(QOS)을 관리할 수 있는 필요한 기능을 갖추어 특정 서비스 수준 계약(SLA)을 충족하는 것이 필수적입니다. 차세대 네트워크를 글로벌 디지털 혁명에 대비하는 것이 중요하다는 점은 이전부터 인정되어 왔으며, 이러한 요구에 부응하기 위한 상당한 진전이 이루어졌습니다. 라디오 세그먼트는 다운링크 중심에서 업링크 중심으로의 전환과 MMWAVE에서 THZ까지의 더 높은 주파수 대역 사용과 함께 새로운 기술로 발전해 왔습니다. 업링크 중심 광대역 통신(UCBC)은 업링크 대역폭을 10배 증가시킵니다. 이러한 유망한 발전은 사물 인터넷(IOT) 개념이 전통적인 센서와 기타 데이터 교환 기술을 넘어, 혁신적인 네트워크 서비스를 용이하게 하기 위해 IIOE[6]에 통합된 상호 연결된 네트워크에 초점을 맞추는 산업용 사물인터넷(IIOE)이라는 새로운 개념으로 전환하여 산업 및 제조업체에 이점을 제공할 수 있습니다 [2], [3]. 무선 부문에서 많은 진전이 있었지만 유선 인프라 부문도 유사하게 발전해야 합니다. 이러한 인프라는 무선 네트워크의 증가하는 용량과 일치해야 하며, 전면, 중간 및 백홀 네트워크 통신에 중점을 두고 네트워크 서비스 제공에서 QOS 관리를 용이하게 하는 메커니즘을 포함해야 합니다. 이에 대응하여 현재 연구에서는 다양한 요구를 가진 서비스 세트를 단일 네트워크 인프라에서 동시에 작동할 수 있게 하는 개념인 트랜스포트 네트워크 슬라이싱을 채택하고 있습니다 [4]. 네트워크 슬라이싱은 5G에 의해 가능해진 주요 기회 중 하나로 널리 인식되었으며, 향후 네트워크 세대에서도 지속적으로 연구될 것으로 예상됩니다 [5], [6]. 네트워크 슬라이싱은 기존 네트워크를 특정 사용 사례 요구 사항을 충족하도록 조정된 다양한 가상 네트워크로 분할하는 데 사용되는 방법입니다. 이러한 분할을 통해 네트워크 운영자는 물리적 인프라를 여러 가상 네트워크(다중 테넌시[7]라고 함)로 분할하여 각각 고유한 전용 리소스와 구성을 할당할 수 있습니다. 네트워크 슬라이싱을 통해 운영자는 네트워크 서비스를 관리하고 제공하는 데 있어 더 큰 적응성, 효율성 향상 및 QOS 개선을 달성할 수 있습니다. 연구에서 제안한 슬라이싱 아키텍처는 여러 개의 분리된 네트워크 슬라이스 간에 네트워크 트래픽을 할당함으로써 네트워크 패킷 처리 및 QOS 관리를 개선하는 혁신적인 프레임워크를 제공합니다. 각 슬라이스는 고유한 네트워크 서비스를 제공하는 독립적인 네트워크 엔터티로 기능하며, 연결된 애플리케이션의 특정 요구 사항을 충족하도록 개별적으로 관리 및 확장할 수 있습니다.

그러나 차별화된 서비스를 가능하게 하는 유망한 기능에도 불구하고 네트워크 슬라이싱은 기본 인프라가 적절하게 업그레이드되고 적절한 패킷 처리 시스템으로 장착되지 않은 경우 저성능 시나리오를 초래할 수 있는 데이터 플레인에서 중요한 문제를 제시합니다. 다중 테넌시를 지원하는 가상화된 네트워크 환경에서 네트워크 서비스는 하나의 패킷을 다른 패킷 내에 캡슐화하여 기본 인프라와 분리되어 있으며(오버레이 네트워크), 네트워크 트래픽 처리 및 분류 작업의 운영 복잡성이 증가합니다. 고속 전송에서 대량의 데이터를 처리하기 위한 시스템 리소스 요구 사항이 높으면 병목 현상이 발생하고 전체 이동통신 네트워크의 성능에 부정적인 영향을 미칠 수 있습니다. 따라서 이러한 문제를 해결하고 가상화된 네트워크 환경에서 안정적이고 효율적인 네트워크 슬라이싱을 보장하기 위한 효과적인 솔루션을 개발해야 합니다. 이러한 과제에 대응하여 두 가지 핵심 개념이 등장했습니다. 첫 번째 개념은 데이터 플레인 프로그래밍 가능성(DPP)으로, 네트워크 관리자에게 스위치 및 라우터와 같은 네트워크 장치의 동작을 특정 애플리케이션 및 워크로드의 요구에 더 적합하게 미세 조정할 수 있는 기능을 제공합니다. 데이터 플레인에 맞춤형 논리를 구현함으로써 네트워크 운영자는 더 빠르고 효율적인 패킷 처리는 물론 네트워크 트래픽에 대한 더 세분화된 제어를 달성할 수 있습니다. 프로그래밍 가능한 데이터 플레인은 필드 프로그래머블 게이트 어레이(FPGA), 스마트 네트워크 인터페이스 카드(SMARTNIC) 또는 소프트웨어 기반 솔루션과 같은 다양한 기술을 사용하여 구현할 수 있습니다. 그러나 데이터 플레인에 맞춤형 논리를 설계하고 구현하려면 전문적인 기술과 전문 지식이 필요하며 네트워크 운영에 추가적인 복잡성을 도입할 수도 있습니다. 두 번째 개념은 네트워크 하드웨어 오프로드로, 호스트의 CPU에 대한 높은 작업 부하의 부정적인 영향을 극복하기 위해 호스트 소프트웨어의 일부 네트워크 관련 작업을 전문 하드웨어 구성 요소에 위임하는 방법입니다. 네트워크 하드웨어 오프로드는 네트워크 관련 작업에 소비되었을 CPU 사이클을 확보합니다. 이를 통해 CPU 사용량이 크게 줄어들어 시스템 성능과 확장성이 향상될 수 있습니다. 이번 연구의 목표는 향후 6G 전면-후방 통신에서 네트워크 슬라이싱을 가속화할 효율적인 패킷 처리 및 분류 솔루션을 개발하는 것입니다. 주요 목표는 향후 6G 사용 사례에 의해 부과된 까다로운 서비스 품질(QOS) 요구 사항

[6] IOE의 산업용 버전

[7] 여러 사용자가 같은 물리적 또는 가상 인프라를 공유하지만, 각각 다른 테넌트로 구분되어 자체적인 데이터와 환경을 유지하는 것

을 충족하는 것입니다. 그러나 이 목표를 달성하기 위해 고려해야 할 다양한 과제가 있습니다.

A. **캡슐화된 네트워크 트래픽 지원** 네트워크 하드웨어 오프로드와 관련하여 일부 네트워크 인터페이스 카드(NIC)는 수신 측 스케일링(RSS) [8]기술을 갖추고 있어 NIC가 들어오는 네트워크 트래픽을 시스템 내의 여러 큐와 해당 처리 코어에 분배할 수 있습니다. NIC는 들어오는 각 네트워크 패킷에 대한 해시 식별자를 생성합니다. 해시는 소스 및 대상 IP 주소, 소스 및 대상 포트, 프로토콜 유형과 같은 여러 패킷 헤더 필드를 기반으로 계산됩니다. 그런 다음 NIC는 이 해시 값을 사용하여 특정 수신 큐에 패킷을 할당하고 이 큐는 특정 CPU 코어와 연결됩니다. RSS는 여러 코어에 워크로드를 분산하여 성능 저하를 방지하고 시스템의 네트워크 처리 기능의 전반적인 효율성을 향상시킬 수 있습니다. 이 연구에서 RSS는 네트워크 슬라이싱을 달성하기 위해 특정 서비스나 애플리케이션과 관련된 네트워크 트래픽을 전용 큐로 전달하는 데도 사용할 수 있습니다. 그러나 많은 네트워크 인터페이스 카드(NIC)에서 사용되는 기본 RSS 알고리즘에는 한계가 있습니다. 이러한 알고리즘은 각 패킷의 몇 가지 헤더 필드만 검사하여 해시 식별자를 생성할 수 있으며 패킷 심층 검사를 수행할 수 없습니다. B5G 및 향후 6G 네트워크에서와 같이 가상화된 멀티 테넌트 시나리오에서는 트래픽이 깊게 캡슐화될 수 있으므로 서로 다른 네트워크 사업자나 서비스 애플리케이션을 구분하기 위해 내부 프로토콜을 탐색해야 합니다. 이로 인해 모든 들어오는 네트워크 패킷이 NIC(동일한 해시 식별자)에 의해 단일 플로우로 간주되어 패킷이 서로 다른 수신 큐와 CPU에 분산되지 않아 병목 현상과 CPU 정체로 이어질 수 있습니다. 따라서 다중 큐 네트워크 카드를 효과적으로 활용하기 위해 캡슐화된 네트워크 트래픽을 검사, 분류 및 처리할 수 있는 알고리즘으로 네트워크 카드를 프로그래밍하는 메커니즘을 마련하는 것이 중요합니다. 이 연구에서는 확장된 버클리 패킷 필터(EBPF)를 활용하여 캡슐화 지원을 갖춘 특수 네트워크 트래픽 필터링 프로그램을 생성하며, 이는 EXPRESS DATA PATH(XDP) 아키텍처를 사용하여 AGILIO CX SMARTNIC에 오프로드할 수 있습니다.

B. 커널 네트워크 스택의 한계 네트워크 하드웨어 오프로드 기술은 CPU의 처리 부하를 어느 정도 줄이고 전체 네트워크 성능을 향상시킬 수 있지만, 여전히 LINUX 네트워크 서브시스템은 패킷을 최종 목적지로 전달하는 책임을 집니다. 패킷이 커널 네트워크 스택에 도착하면 프로토콜 처리 및 소켓 버퍼링을 포함한 여러 처리 단계를 거칩니다. 여기서 잠재적인 병목 현상은 들어오는 패킷이 애플리케이션에 의해 처리되기 전에 버퍼에 저장되는 소켓 버퍼링 단계에서 발생할 수 있습니다. 버퍼 크기가 제대로 구성되지 않았거나 애플리케이션이 패킷을 충분히 빠르게 처리할 수 없는 경우, 패킷이 손실되고 네트워크 성능이 저하될 수 있습니다. 이러한 문제는 이전 연구에서 시스템의 네트워크 스택이 높은 속도의 패킷 처리를 감당하지 못해 다른 서버로 패킷을 전달해 처리하는 방법으로 극복해야 했을 때 관찰되었습니다. 본 연구에서는 NETRONOME FLOW PROCESSOR(NFP) 드라이버를 위한 XDP 기반 커널 바이패스 기술(AF XDP)을 개발하고 통합하며 평가했습니다. 이 접근 방식은 이미 하드웨어 수준에서 처리된 네트워크 패킷을 드라이버가 커널 네트워크 서브시스템에서 발생하는 처리 부담 없이 최종 목적지로 전달할 수 있도록 합니다. 드라이버 수준에서 커널 바이패스를 지원하는 다른 SMART NIC도 있지만, 우리의 지식으로는 동일한 SMARTNIC 아키텍처에서 XDP 기반 기술, 하드웨어 오프로드 및 드라이버 AF XDP를 결합하여 고성능 패킷 처리 및 전송 네트워크 슬라이싱 기능을 달성한 것은 처음입니다.

C. 차세대 네트워크 호환성 앞서 언급했듯이 데이터 플레인을 프로그래밍하는 것은 네트워크 관리자에게 전문적인 기술을 요구하는 복잡한 작업이며, 이를 통해 시스템에 추가되는 각 애플리케이션에 대한 특정 기능을 갖춘 맞춤형 구성 프로그램을 만들어야 합니다. 이 과정은 네트워크 운영에 지연을 초래할 수 있습니다. 반면에 NGN(차세대 네트워크)은 소프트웨어 정의 네트워킹(SDN) 또는 네트워크 기능 가상화(NFV)와 같은 첨단 기술을 통해 동적 자원 할당 및 주문형 서비스 제공을 위해 설계되었습니다. 따라서 NGN의 현재 접근 방식에 부합하려면 유연성과 주문형 기능을 지원하는 솔루션을 개발해야 합니다. 이 연구에서는 데이터 플레인 프로그래밍을 단순화하는 사용자 친화적인 DPP 네트워크 애플리케이션을 제안합니다. 이 애플리케이션은 제어 및 모니터링 기능을 모두 제공하며 쉽게 주문형으로 배포할 수 있습니다. 유연한 구성은 다양한 사용 사례의 요구를 충족하도록 조정할 수 있습니다. 데이터 플레인 프로그래밍의 복잡성을 숨김으로써 이 애플리케이션은 네트워크 관리 프로세스를 단순화하여 다양한 수준의 전문 지식을 가진 사용자가 더 쉽게 접근할 수 있도록 합니다.

D. 다양한 시나리오에 적응하는 유연성 이 연구에서는 현재 5G 기술과 호환되는 솔루션을 제안하지만, 향후 6G 사용 사

[8] NIC 에서 수신한 네트워크 트래픽을 처리하기 위한 기술, 병목현상 방지, 네트워크 트래픽 처리 속도를 향상시킴

례는 초저지연, 높은 신뢰성 및 대규모 연결이 필요하기 때문에 더욱 높은 핵심 성과 지표(KPI)를 요구할 것으로 예상됩니다. SAAD 등은 6G 네트워크가 기본 서비스에 의해 주도되는 기술 동향을 통합할 것이며, 따라서 5G 및 B5G(5G 이후) 네트워크에 비해 향상된 요구 사항 및 KPI가 필요하다고 강조했습니다. 이러한 사용 사례에는 실시간 및 미션 크리티컬 애플리케이션인 자율 주행, 스마트 공장 및 원격 수술 등이 포함되어 많은 데이터와 복잡한 네트워크 토폴로지(네트워크 내의 다양한 요소가 서로 어떻게 연결되어 있는지를 나타내는 구성)를 필요로 하며 KPI를 효과적으로 충족하는 데 상당한 과제를 제시합니다. 미래 6G 전면에서 백홀 네트워크에서 과부하 상황을 처리할 수 있도록 하기 위해, 특히 INDUSTRY 4.0(I4.0) 서비스와 관련된 PRE-6G 사용 사례를 통해 실증적 검증을 수행했습니다.

이 논문은 EBPF-XDP, AF XDP 및 EBPF-XDP 하드웨어 오프로드 기술을 모두 결합하여 AGILIO CX SMARTNIC이 NGN 아키텍처의 네트워크 성능을 크게 향상시키는 동시에 유연한 방식으로 네트워크 트래픽의 선택적 맞춤화 및 분류를 가능하게 하는 방법을 보여줍니다. 구체적으로 이 작업은 이러한 기술을 활용하여 클라우드 서비스 제공업체 내부의 유선 네트워크 세그먼트를 가속화하여 다가오는 6G 네트워크에서 네트워크 통신을 가속화하고 CPU에 부담을 주지 않고 시스템의 리소스를 최대화하여 네트워크 처리 작업을 최적화하는 것을 목표로 합니다. 이 작업은 향후 6G 네트워크에서 I4.0의 까다로운 요구 사항을 충족하는 솔루션을 제공하고 이 솔루션이 성공적으로 적용될 수 있는 몇 가지 사용 사례를 제시합니다. 이를 위해 제안된 솔루션은 집중적인 실증 검증을 수행하여 이 작업의 타당성을 입증하고 향후 6G 네트워크에서 예상되는 까다로운 KPI를 충족시킵니다.

본론

이 섹션에서는 관련 문헌에 대한 종합적인 검토를 제공하며, 특히 패킷 처리 기술에 중점을 둡니다. 구체적으로, 그리고 이 연구의 범위와 일치하여, 이 검토는 이러한 기술과 구현을 커널 바이패스 및 하드웨어 오프로드 기술의 두 가지 주요 그룹으로 분류하여 각각에 대한 심층 분석을 제공합니다.

A. 커널 바이패스 기술 시스템 커널과 사용자 공간 애플리케이션 간의 인터페이스는 일반적으로 성능 병목 현상의 주요 원인입니다. 순수한 LINUX는 약 1MPPS를 효율적으로 처리할 수 있을 뿐이며, 이는 최소 처리량이 10MPPS인 예상 6G KPI와는 거리가 멉니다【8】. 커널 바이패스 기술은 커널 네트워크 스택에서 발생하는 비용이 많이 드는 컨텍스트 전환의 사용을 피함으로써 이를 해결하는 것을 목표로 합니다. 가장 인기 있는 커널 바이패스 솔루션 중 하나는 DPDK(데이터 플레인 개발 키트)【9】로, 광범위한 CPU 아키텍처에서 사용할 수 있는 패킷 처리 작업을 가속화하기 위한 라이브러리 세트로 구성되어 있습니다. 이 솔루션에서는 사용자 공간 애플리케이션이 네트워킹 하드웨어를 제어하는 반면 운영 체제(OS)는 우회됩니다. SNABBSWITCH【10】는 LUA 언어로 작성된 네트워킹 프레임워크로 주로 L2 애플리케이션을 작성하기 위해 사용됩니다. 이 프레임워크는 UIO(UNIVERSAL INPUT OUTPUT)에 의존하는 전체 프레임워크라는 점에서 DPDK와 매우 유사합니다. NETMAP【11】은 UIO 기술을 사용하여 구현되지는 않지만 커널 모듈을 사용합니다. 그러나 네트워킹 하드웨어와 통합되어 있지 않으며 드라이버 커스터마이징이 필요합니다. PACKET MMAP【12】은 엄밀히 말하면 커널 바이패스 기술은 아니지만 매우 제한된 버퍼와 하나의 시스템 호출을 사용하여 네트워크 트래픽을 효율적으로 캡처하여 패킷의 원시 전송 효율성을 향상시키는 데 사용됩니다. PF RING【13】은 패킷 캡처 속도를 높일 수 있는 네트워크 소켓 유형입니다. 이 도구는 LINUX NAPI를 사용하여 NIC에서 패킷을 폴링하여 네트워킹 장치에 대한 인터럽트 완화를 가능하게 합니다. EF VI【14】를 사용하면 사용자가 상위 계층 프로토콜을 직접 구현해야 하지만, 사용자 수준에서 원시 이더넷 프레임을 직접 송수신하여 CPU 오버헤드를 줄이고 성능을 높일 수 있습니다. AF XDP【15】는 NIC에서 사용자 애플리케이션 계층으로 원시 네트워크 프레임을 전송하기 위해 XSK 소켓을 사용하며 고성능 패킷 처리를 위해 최적화되어 있습니다. 이 도구에서 각 소켓은 RX 및 TX 링과 연관됩니다. 패킷을 전송하려면 이 링들이 UMEM이라고 불리는 메모리 영역의 데이터 버퍼를 사용하며 이는 크기가 같은 프레임으로 나뉜 가상 연속 메모리 영역으로 설명됩니다. 이전 솔루션과 달리 AF XDP는 기본 LINUX 커널의 일부인 EBPF-XDP와 통합되어 있습니다. 또한 EBPF-XDP는 하드웨어 오프로드 또는 네트워크 드라이버와 같은 다양한 컨텍스트에서 데이터 플레인을 프로그래밍할 수 있도록 합니다. 그러나 우리가 아는 한, 현재 EBPF-XDP 하드웨어 오프로드를 지원하는 유일한 SMARTNICS인 NETRONOME의 AGILIO SMARTNICS는 AF XDP를 지원하지 않습니다. 따라서 AF XDP를 통해 드라이버에서 하드웨어 오프로드 및 커널 바이패스의 두 구현을 모두 결합하는 것은 불가능했습니다. 본 연구의 목적을 위해 이 SMARTNIC의 드라이버는 필요한 AF XDP 기능을 통합하도록 확장되었습니다.

B. 하드웨어 오프로드 기술 프로그래밍 가능한 하드웨어는 메모리에 지속적인 액세스를 제공하므로 필터링 규칙의 수가 증가하더라도 추가 메모리 액세스 시간 오버헤드가 발생하지 않으며 더 구체적인 대기 시간 제어를 할 수 있습니다. 더욱이, 하드웨어 장치에는 소프트웨어 데이터 플레인보다 우수한 처리 속도가 있어 더 빠른 전송 속도를 암시합니다. 최근 몇 년 동안 XILINX는 인공 지능, 빅 데이터, 분석, 하이퍼스케일, 머신러닝, 스토리지 및 텔코 애플리케이션을 포함하여 거

래 환경 및 엔터프라이즈 데이터 센터 모두에 고성능 기능을 갖춘 다양한 FPGA 및 SMARTNIC을 제공했습니다【16】【17】【18】【19】【20】【21】. 이 카드들은 최대 30TB/S의 내부 파이프라인 SRAM 대역폭과 최대 100GBE의 물리적 네트워크 인터페이스를 갖춘 최신 데이터 센터의 까다로운 요구 사항을 충족하도록 설계되었습니다. 그러나 이러한 카드에는 두 가지 중요한 문제가 있습니다. 첫 번째는 높은 가격이고 두 번째는 소프트웨어 개발 키트(SDK) 및 드라이버의 사유 소유입니다. INTEL은 가상화된 무선 액세스 네트워크(VRAN) 솔루션을 포함하여 NFV 인프라를 위한 고성능 및 저지연 연결을 제공하기 위해 SMARTNIC 솔루션을 도입했습니다【22】【23】【24】. 그러나 높은 비용과 독점 소스 코드 때문에 네트워크 관리자는 네트워킹 분야의 학계가 가장 매력적인 옵션으로 간주하지 않을 수 있습니다. NETFPGA【25】는 최대 100GBPS의 네트워크 전송을 제공할 수 있는 오픈소스, 라인 속도 및 유연한 플랫폼입니다. 또한 이 카드는 P4【26】또는 VHDL【27】을 사용하여 프로그래밍할 수 있습니다. 그러나 SUME RIFFA 드라이버 단점으로 인해 NETFPGA 기반 솔루션이 달성한 성능은 향후 6G 네트워크 KPI를 충족하지 못한다는 것이 【28】【29】【30】【31】에서 입증되었습니다. NETRONOME【32】과 CORIGINE【33】은 오버레이, 원격 측정 또는 보안과 같은 기능을 포함하여 서버 네트워킹을 위한 뛰어난 효율성을 제공하는 AGILIO SMARTNIC 및 소프트웨어를 제공합니다. NETRONOME과 CORIGINE은 SMARTNIC을 P4 또는 제한 C 언어를 사용하여 프로그래밍할 수 있도록 하는 SDK를 제공합니다. 이러한 SMARTNIC는 합리적인 가격이며 펌웨어 및 NFP 드라이버가 오픈소스입니다. 또한 이 하드웨어는 EBPF-XDP 하드웨어 오프로드를 지원하여 하드웨어 프로그래밍 가능성과 유연성 측면에서 매우 바람직합니다. NFP 드라이버는 LINUX 커널을 우회하는 진정한 AF XDP 지원을 제공하지 않지만, NETRONOME 및 CORIGINE과 협력하여 이 기능이 NFP 드라이버에 통합되었습니다. 그 결과, 여기에서는 PRE-6G 시나리오에서 복잡한 오버레이 네트워크를 처리하기 위해 하드웨어 오프로드 및 커널 바이패스(AF XDP를 통해) 기술을 결합한 새로운 접근 방식을 제안합니다.

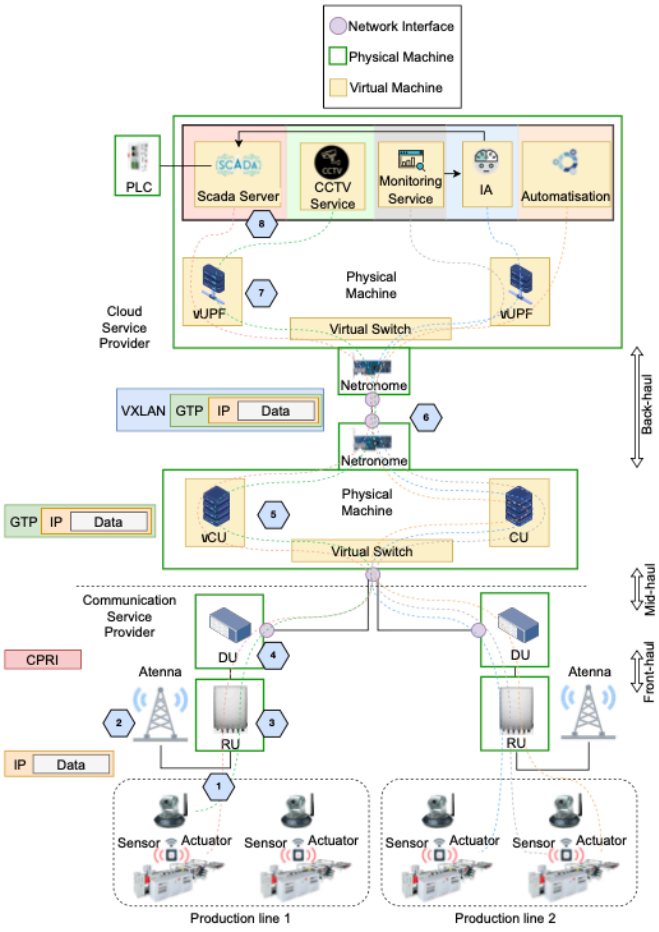


FIGURE 1 <5G 및 6G를 넘는 멀티 테넌트 아키텍처의 프로토타입>

III. 다중 테넌트 프리-6G 아키텍처 개요 FIGURE 1 은 인더스트리 4.0(I4.0) 환경에서 5G 이후 및 프리-6G 아키텍처를 보여줍니다. 다음 설명의 가독성을 높이기 위해 이 그림 전반에 걸쳐 다양한 숫자 레이블이 삽입되었습니다. 레이블 1은 여러 생산 라인이 통신 서비스 제공업체(CSP)에 연결되어 있음을 보여줍니다. 이러한 생산 라인은 서로 다른 가상 이동통신 사업자에 속합니다. CSP는 다중 입력 다중 출력(MIMO) 안테나, 라디오 유닛(RU), 분산 유닛(DU)의 사용을 통해 센서와 액추에이터 간의 무선 통신을 용이하게 하는데, 이는 그림 1에서 각각 레이블 2, 3, 4로 표시됩니다. RU는 디지털 데이터를 무선으로 공기를 통해 사용자 장치로 전송할 수 있는 무선 신호로 변환하는 역할을 합니다. DU는 RU 가까이에 배치된 물리적 구성 요소로, 라디오 링크 제어(RLC), MAC 및 물리 계층의 일부 부분을 실행하며 중앙 집중식 유닛(CU)에서 제어합니다. RU에서 DU로, DU에서 가상 CU(vCU)로, 그리고 vCU에서 코어 네트워크로의 연결은 각각 프론트홀, 미드홀 및 백홀이라고 합니다.

아키텍처의 핵심은 물리적 컴퓨터가 다양한 가상 네트워크 운영자 간에 공유되는 분산형 가상 네트워크 아키텍처를 제공하는 클라우드 서비스 제공업체에서 제어합니다. 클라우드 서비스 제공업체 세그먼트에는 가상화된 CU(vCU), 가상화된 사용자 평면 기능(vUPF) 및 수직 서비스와 같은 다양한 구성 요소가 포함되며, 이러한 구성 요소는 그림 1에서 각각 레이블 5와 7로 표시됩니다. vCU를 통해 사용자는 통신 서비스 제공업체와 클라우드 간 통신이 가능하며, 이를 통해 사용자는 코어 네트워크 세그먼트에 액세스할 수 있습니다. vUPF는 네트워크 데이터 경로를 통해 사용자 데이터를 전달하는 것 외에도 사용자 이동성, 인증, 핸드오버 관리, 사용자 등록 등을 담당합니다**. vUPF는 vCU에서 다양한 수직 서비스(레이블 8에서 볼 수 있는)로 사용자 트래픽을 전달**하며, 이는 다양한 사용자 요구와 고급 네트워크 슬라이싱 정책의 구현을 통해 충족되는 까다로운 SLA를 가지고 있습니다. 설명된 각 구성 요소의 작업에 대한 자세한 설명은 [34]에서 자세히 제공됩니다.

도시된 아키텍처는 프리-6G 인프라의 하드웨어 및 소프트웨어 구성 요소 간에 고성능 연결 및 고급 네트워크 슬라이싱 지원을 달성하기 위한 목표로 분리된 포워딩 장치를 기반으로 하는 하이브리드 소프트웨어/하드웨어 접근 방식을 제시합니다. 그림 1의 **레이블 6은 vCU와 vUPF 간의 네트워크 트래픽 전달을 담당하는 SmartNIC를 식별**합니다. 이 SmartNIC는 네트워크 데이터 경로 처리, 제어 및 모니터링 기능을 오프로드하여 프리-6G 네트워크를 지원하는 슬라이싱 친화적인 환경을 조성합니다. 이를 통해 소프트웨어 구성 요소에서 제공하는 유연성을 유지하면서 네트워크 통신 성능을 최적화하는 것이 목표입니다.

제시된 인프라는 하드웨어 기반 제어 및 모니터링 기능이 있는 유연한 데이터 경로와 커널 바이패스 메커니즘을 사용하는 소프트웨어 소켓을 제공하여 하드웨어 장치에서 네트워크 기능 가상화(NFV)까지 네트워크 트래픽 가속을 허용합니다. 이 논문은 6G 고급 네트워크 슬라이싱을 지원하고 클라우드 서비스 제공업체를 위해 설계된 일반 스토리지를 오프로드하기 위해 Agilio CX SmartNIC 기반 고성능 제어 및 모니터링 프로그래밍 가능 데이터 경로를 사용하며 6G 네트워크 수직 사용 사례가 부과한 까다로운 KPI를 충족하는 것을 목표로 합니다. 클라우드 서비스 제공업체 네트워크 세그먼트에서 네트워크 패킷은 센서와 액추에이터에서 수신 및 전송되며 각각 CU에 의해 GPRS(일반 패킷 라디오 서비스) 터널링 프로토콜(GTP)을 사용하여 캡슐화됩니다. 이 프로토콜은 다양한 위치를 따라 네트워크 이동성을 허용합니다.

TABLE I: Use cases key performance requirements

Use case	E2E Latency	Reliability	Bandwidth	Packet Size	Slice config
(1)	<0.1ms	10^{-8}	<100Mbps	<1500B	<1s
(2)	<33ms	10^{-6}	<3Gbps	<1500B	<1s
(3)	<0.15ms	10^{-6}	<10Gbps	<300B	<1s
(4)	<0.5ms	10^{-5}	<10Gbps	<132B	<1s
(5)	<0.1ms	10^{-6}	<10Gbps	>200B	<1s

또한 가상 스위치는 네트워크 트래픽을 가상 네트워크 운영자 간에 격리하고 네트워크 리소스를 논리적으로 분리할 수 있도록 두 번째 터널을 생성합니다. 이러한 터널을 생성하는 데 사용할 수 있는 다양한 네트워크 프로토콜이 있지만, 이 출판물에서 수행된 실험을 위해 VXLAN(Virtual Extensible LAN)이 특별히 선택되었습니다. 이 원고에 개략적으로 설명된 제안된 아키텍처는 AF XDP 및 Agilio CX를 사용하여 다양한 논리 경로를 사용할 수 있다는 점을 포함하여 여러 이점을

제공합니다. 또한 미래 6G 네트워크에서 요구되는 까다로운 KPI를 충족하면서 실시간 모니터링 지표를 제공합니다.

IV. 프리-6G 사용 사례 이 연구는 동적 제어, 모니터링 및 네트워크 슬라이싱이 가능한 고성능 프로그래밍 가능한 데이터 플레인을 개발하는 것을 목표로 합니다. 이를 실현 가능성을 검증하기 위해, 특히 인더스트리 4.0 환경과 관련된 일련의 까다롭고 새로운 사용 사례가 선정되었습니다. 이 검증을 통해 데이터 지연 시간, 안정성 및 대역폭과 같은 KPI가 신중하게 고려될 것입니다. 목적은 미래의 6G 네트워크의 데이터 플레인(프론트홀에서 백홀까지)이 필요한 성능과 효율성으로 이러한 고급 사용 사례를 지원할 수 있도록 하는 것입니다. 이러한 사용 사례는 6G-BRAINS 프로젝트의 딜리버러블 2.1에서 사용할 수 있는 사용 사례를 기반으로 하며 해당 요구 사항은 표 I에 나열되어 있습니다.

A. 사용 사례(1): 프로그래머블 로직 컨트롤러(PLC) 제어 기능을 엣지로 오프로드 PLC와 같은 고정형 기계 컨트롤러는 공장에 관련된 소프트웨어 및 하드웨어 구성 요소의 유연성을 제공하기 위해 가상화된 환경에서 실행되는 소프트 컨트롤러로 대체될 예정입니다. 새로운 가상화된 구성 요소와 기계 구성 요소 간의 통신 네트워크는 6G 시나리오와 관련된 성공적인 트래픽 전송 및 제어 방법을 위한 중요한 요소입니다. 이러한 기능을 컨테이너 또는 가상 머신으로 실행하여 엣지에 PLC 제어 기능을 오프로드하는 것은 인더스트리 4.0에서 생산 프로세스를 제어하는 데 더 큰 유연성을 제공합니다. 가상화된 컨트롤러와 기계 구성 요소 간의 통신에는 네트워크 슬라이싱 전략을 적용하여 낮은 산업용 애플리케이션 주기 시간과 매우 정밀한 동기화를 지원해야 하는 결정론적 통신을 보장하기 위해 저지연 및 고신뢰성이 필요합니다. 이 사용 사례와 관련된 주요 성능 요구 사항은 표 I의 행(1)에 설명되어 있습니다.

B. 사용 사례(2): 스마트 운송 차량: 위치 추적 및 비디오 처리 오프로드 AGV(Automated Guided Vehicle)는 인더스트리 4.0의 핵심 기여자입니다. 최첨단 공장은 여전히 수작업 또는 반자동 기계에 기반을 두고 있지만, 미래의 공장은 새로운 AI 기반 AGV에 투자하고 있습니다. 이 접근 방식은 회사의 수익에 긍정적인 전망과 연관된 전체 공장 성능의 증가를 수반합니다. AGV는 운전자 없이 제품과 재료를 생산 라인으로 이송하는 등의 작업을 처리하는 인더스트리 4.0의 큰 자산입니다. 또한 이러한 차량은 AGV에 장착된 센서를 통해 그리퍼 암을 사용하여 부품을 집고 배치하여 조립 프로세스를 용이하게 하기 위해 로봇 장비를 운반할 수 있습니다. 이는 초기 및 반복 노력이 거의 없이 AGV가 공장 효율성을 크게 향상시킨다는 것을 의미합니다.

이 사용 사례는 두 가지 중요한 AGV 기능으로 나눌 수 있습니다. a) AGV 내장 카메라를 사용하여 비디오를 캡처하고 객체 감지 및 의사 결정용으로 엣지 컴퓨팅 노드로 비디오를 스트리밍합니다. b) 실시간 의사 결정 알고리즘을 기반으로 실내 및 실외 자율 내비게이션이 있습니다. 고품질 비디오 카메라는 AGV에서 6G 네트워크 아키텍처의 엣지 노드로 비디오를 전송하는 데 사용되며, 여기서 비디오를 분석하고 의사 결정을 내립니다. 이 비디오 전송에는 높은 데이터 속도 요구 사항을 보장하는 최적의 통신을 보장하는 효율적인 네트워크 아키텍처가 필요하며, 약 130개의 비압축 HD 프레임 전송에서 카메라당 최대 3Gbps를 소비합니다. 이 사용 사례에서 데이터 속도 요구 사항, 초고정밀 및 의사 결정은 표 I의 행(2)에 설명된 KPI를 충족하기 위해 현재의 첨단 기술과 관련하여 분석됩니다.

C. 사용 사례(3): 고급 네트워크 슬라이싱 이전에 설명한 사용 사례는 대부분 단일 유형의 트래픽을 나타내며 애플리케이션 통신과 관리 및 제어 트래픽을 격리하는 것 외에는 엄격하게 요구되는 우선 순위 지정 기술이 없습니다. 그러나 동일한 환경에서 다양한 애플리케이션의 통신 리소스를 공유하는 것은 일반적입니다. 일부 인더스트리 4.0 환경에서는 사용 사례 간의 요구 사항이 크게 다를 수 있으므로 다양한 사용 사례의 KPI를 효율적으로 보장하기 위해 고급 네트워크 슬라이싱 기술을 구축해야 합니다. 이러한 기술은 매우 역동적이고 이질적인 트래픽 제어 요구 사항을 해결하고 또한 인더스트리 4.0을 위한 실시간 QoS를 보장합니다.

고급 네트워크 슬라이싱은 다양한 고객 요구 사항에 대해 유연하고 맞춤화된 E2E 보장된 QoS를 제공합니다. 네트워크 데이터 통신을 기반으로 여러 세분성 수준의 고도로 유연한 네트워크 슬라이스를 주문형으로 정의할 수 있습니다. 고가용성과 효율성을 확보하려면 네트워크 슬라이스 인스턴스화 시간이 1초 미만이어야 합니다. 동일한 네트워크에서 다양한 사용 사례가 공존하는 복잡한 산업 환경에서 의도 기반 네트워크 슬라이싱 관리는 관리 작업을 용이하게 하고 독립적인 네트워크 트래픽 통신에 대한 해당 QoS를 보장합니다. 이를 통해 6G 네트워크 아키텍처를 사용하여 복잡하고 중요한 인더스트리 4.0 시나리오에서 SLA를 준수할 수 있습니다. 고급 네트워크 슬라이싱의 까다로운 요구 사항은 표 I의 행(3)에 설명되어 있습니다.

D. 사용 사례 (4): 실내 농업 시나리오에서 동물 추적 스마트 농업은 전통적인 농장에서 IoT 기반 농업으로 진화하는 것으로, 현대의 정보 및 통신 기술을 사용하여 생산의 품질과 양을 모두 증가시키는 동시에 인간의 상호 작용을 최적 수준으로 줄이는 것을 의미합니다. 이러한 최적화는 다양한 측면에서 수행될 수 있지만, 이 사용 사례는 실내 농업 시나리오에서 동물 추적에 중점을 둡니다. 소, 돼지 또는 닭과 같은 농장 동물은 동물의 활동, 복지 및 건강을 모니터링하는 목걸이를 착용하며 동물 복지에 영향을 미칠 수 있는 것이 감지되면 농부와 규제 기관에 즉시 알립니다. 모니터링 데이터는 엣지 컴퓨터 노드로 전송되어 AI 에이전트가 동물의 이동성과 건강 데이터를 처리하여 동물의 복지와 건강을 보장합니다. 이 사용 사례와 관련된 KPI는 표 I, 행 (4)에 설명되어 있습니다.

E. 사용 사례 (5): 공항 서비스 및 수하물 처리 로봇 이 사용 사례는 공항에서 수하물 처리 컨베이어 벨트 위치 간 승객 수하물을 운반하기 위한 자동 유도 차량(AGV) 로봇을 사용하여 6G 자동 수하물 처리 시스템을 제시합니다. 서비스 로봇과 AGV 경로의 물체에 대한 비디오 라이브 스트리밍을 사용하여 경로의 물체에 대한 객체 인식 및 충돌 방지를 수행합니다. 승객이 공항에 도착하면 수하물은 체크인 라인에 내려지고 AGV는 방금 셀프 체크인한 수하물을 가져와 도착 항공기로 운송하기 위해 올바른 컨베이어 벨트로 옮겨 최소한의 수하물 처리 직원으로 자동화되고 지속 가능한 수하물 분류 시스템을 만듭니다. 이 사용 사례를 수행하려면 라이브 스트리밍 통신과 실시간 AI 감지 및 실행을 목적으로 고신뢰성 및 저지연 통신이 필요합니다. 이 사용 사례에서 필요한 까다로운 통신 요구 사항은 표 I, 행(5)에 설명되어 있습니다.

V. 프로토타이핑 플랫폼 및 구현

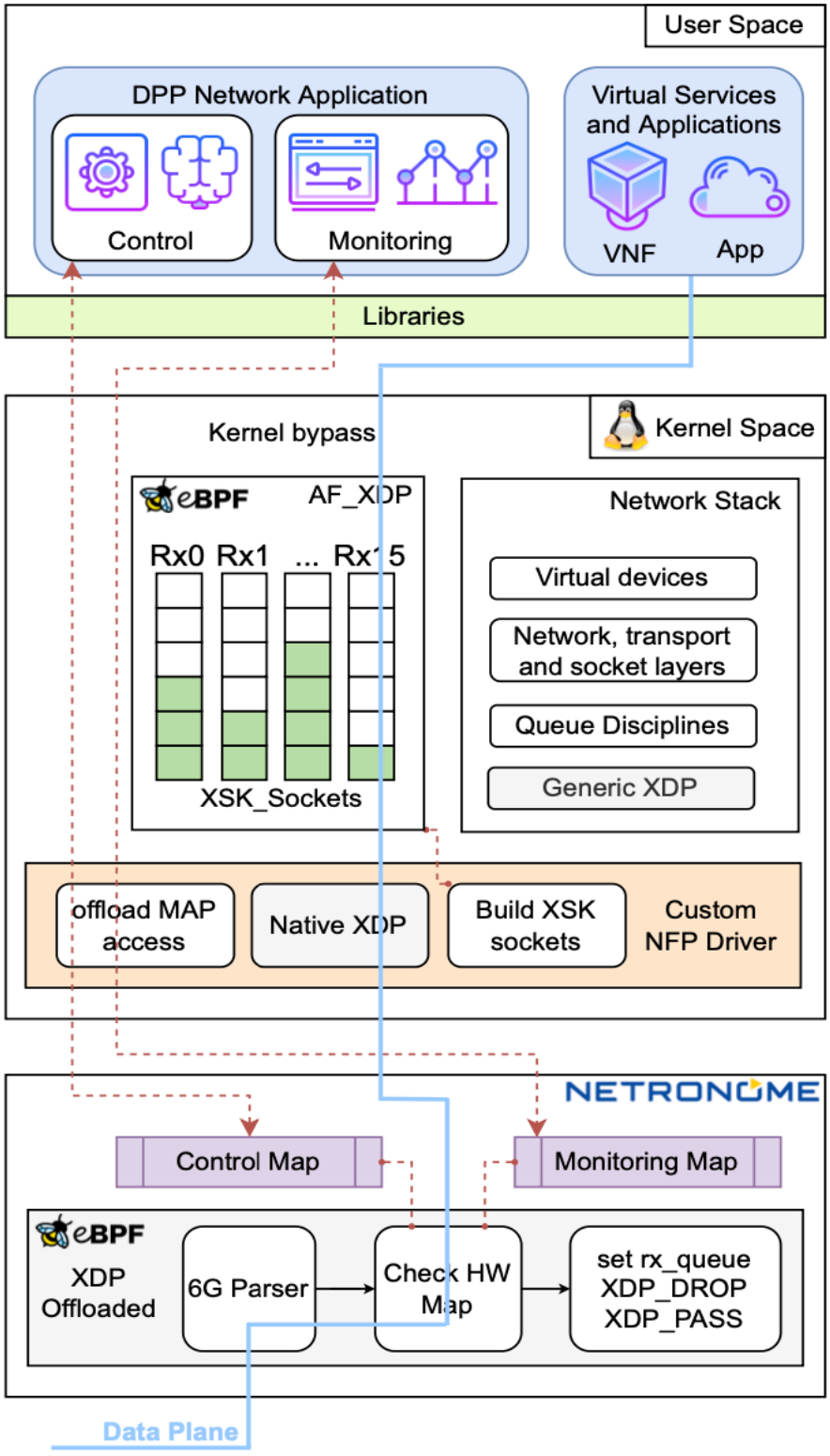
다음 하위 섹션에서는 6G 프론트홀에서 백홀 네트워크에서 고성능 및 고신뢰성 통신을 보장하기 위해 구현된 제안된 프레임워크 아키텍처, 구현된 알고리즘 및 프로토타이핑 구현에 대한 자세한 설명을 제공합니다.

A. XDP 기반 Agilio CX SmartNIC 참조 데이터 경로 이 연구는 최대 처리 속도 25Gbps의 Netronome Flow Processor(NFP) 드라이버를 통해 XDP 및 DPDK와 광범위하게 통합된 투명한 오프로드 SmartNIC를 제공하는 Agilio CX SmartNIC 플랫폼을 기반으로 합니다. 제안된 솔루션은 XDP, eBPF 및 NFP 드라이버의 맞춤형 버전을 사용하여 생성되었습니다. Agilio CX SmatNIC에는 NFP-4000 프로세서가 있으며 60개의 프로그래밍 가능한 흐름 처리 코어, 48개의 패킷 처리 코어, RX 및 TX 채널이 있는 PCIe Gen3 2x8 인터페이스 및 2xSFP28 25GbE 물리적 인터페이스도 포함합니다. SmartNIC는 Agilio eBPF 펌웨어를 사용하여 XDP를 오프로드 및 드라이버 수준에서 구현합니다. 구현된 솔루션은 최대 16개의 하드웨어 큐로 작동하며, 각 하드웨어 큐는 카드에서 사용자 공간으로 트래픽을 전달하는 XSK 소켓에 직접 연결되어 있습니다. 이를 구현하기 위해 Netronome 및 Coregine에서 제공한 NFP 드라이버의 맞춤형 버전이 이 연구의 범위에 맞게 특별히 사용되었으며 AF XDP 지원을 제공합니다. AF XDP는 XSK 소켓을 만들어 고성능 패킷 처리를 최적화한 주소 패밀리입니다【15】. 이 구현의 새로움은 (1) 다중 테넌시 지원을 통해 구현된 오프로드 및 프로그래밍 가능한 프리-6G 데이터 경로, (2) 오프로드된 고성능 실시간 모니터링 도구, (3) 고급 네트워크 슬라이싱 및 트래픽 필터링을 제공하는 오프로드된 제어 기능, (4) XSK 소켓 지원을 제공하고 따라서 Linux 커널 기반 구현의 성능을 향상시키는 NFP 드라이버의 맞춤형 버전에서 비롯됩니다. 우리가 아는 한, 프리-6G 네트워크 아키텍처의 전송 네트워크 세그먼트에서 고성능 통신을 향상시키기 위해 동일한 네트워크 카드를 사용하여 하드웨어 오프로드 기능과 커널 바이패스 기술을 결합한 솔루션을 처음으로 접근한 것입니다.

B. 제안된 프레임워크 아키텍처 그림 2는 이 작업에서 구현된 프레임워크의 내부 아키텍처를 나타냅니다. 이 프레임워크는 Agilio CX SmartNIC을 사용하는 하드웨어, 하드웨어 및 소프트웨어 인터럽트를 처리하는 Linux 커널 공간, 최종 사용자에 가장 가까운 부분인 사용자 공간의 세 부분으로 나뉩니다. 이 그림에서 점선 빨간색 선은 제어 플레인을 나타내고 파란색 선은 데이터 플레인을 나타냅니다.

SmartNIC에는 맞춤형 XDP 오프로드 펌웨어와 제어용 및 모니터링용의 두 가지 다른 eBPF 하드웨어 맵이 있습니다. 이러한 키/값 맵은 각각 200만 개 이상의 규칙을 할당할 수 있으며 일정한 메모리 액세스 시간을 가지고 있습니다. 제어 맵에서 각 규칙에는 32비트 키와 32비트 값이 포함되어 있으며, 이는 동작을 나타내기 위해 16비트와 필요한 경우 동작의 출력 값을 나타내기 위해 16비트로 나뉩니다(예: 특정 rx 큐에 설정). 모니터링 맵의 각 규칙에는 32비트 키와 64비트 값도 포함되어 있으며, 32비트는 패킷 길이, 32비트는 이 특정 규칙에 일치하는 패킷 수를 나타냅니다. 그림에서 회색으로 표시된 XDP 오프로드 펌웨어는 세 가지 다른 구성 요소, 프리-6G 파서, 하드웨어 맵이 확인되고 출력 결정이 이루어지는 논리로 나눌 수 있습니다. 프리-6G 파서는 네트워크 플로우를 처리하고 필요한 메타데이터를 추출하는 역할을 합니다. 이 정보는 나중에 일치 모듈에서 사용할 해시 식별자를 계산하는 데 사용됩니다. 일치 모듈은 네트워크 패킷에 대해 적절한 조치를 취할지 여부를 결정하기 위해 ePBF 제어 맵에 저장된 것과 해시 식별자를 비교합니다. 비교 결과에 따라 일치 모듈은 세 가지 동작 중 하나를 취할 수 있습니다. 특정 하드웨어 큐로 네트워크 패킷을 전달하거나 패킷을 삭제하거나 기

본 큐를 통해 패킷을 커널 네트워크 스택으로 전달할 수 있습니다.



<FIGURE 2 제안된 프레임워크 아키텍처>

네트워크 패킷이 커널 공간에 도달하면 드라이버 수준에서 소켓 전달을 허용하는 NFP 드라이버의 맞춤형 버전을 포함하는 네트워크 스택에 의해 처리됩니다. 일반적으로 공개 NFP 드라이버는 네트워크 스택을 사용하여 하드웨어 카드에서 사용자 공간으로 데이터를 전송합니다. 그러나 네트워크 스택을 통해 데이터를 전송하기 위해 RSS 큐 규율을 사용했던 [7]에서 얻은 결과를 분석한 후, 이 솔루션의 성능이 미래 6G 네트워크의 KPI를 충족하지 못한다는 것이 분명해졌습니다. 네트워크 스택의 병목 현상이 이러한 성능 저하의 원인으로 확인되었습니다. 이 문제를 해결하기 위해 연구팀은 새로운 맞춤형 버전의 NFP 드라이버를 제안했습니다. 이 드라이버는 AF XDP 및 XSK 소켓을 활용하여 네트워크 스택을 우회하고 SmartNIC과 사용자 공간 간의 통신을 가속화합니다. 맞춤형 드라이버 구현은 최대 16개의 소켓을 지원하며 하드웨어 제어 및 모니터링 맵에 대한 액세스를 허용합니다. 또한 Native XDP 지원을 통해 유연한 데이터 경로를 제공합니다. 이 맞춤형 드라이버를 사용함으로써 연구팀은 보다 일반적인 드라이버에 비해 향상된 성능과 효율성을 달성할 수 있었으며, 이를 통해 데이터 플레인이 6G 네트워크의 까다로운 요구 사항을 충족할 수 있게 되었습니다.

사용자 공간 수준에서 두 가지 주요 구성 요소가 있습니다. 데이터 플레인 프로그래밍 가능성(DPP) 네트워크 애플리케이션과 가상 서비스 및 애플리케이션입니다. DPP 네트워크 애플리케이션은 제어 및 모니터링 도구를 제공하는 반면 가상 서비스 및 애플리케이션은 네트워크 데이터 플레인에 포함된 다양한 네트워크 서비스 및 애플리케이션으로 구성됩니다. Libbpf【36】 및 XSK【37】와 같은 여러 라이브러리가 이러한 사용자 애플리케이션과 커널 공간 간의 통신을 용이하게 합니다. 제어 애플리케이션은 하드웨어 제어 맵에 네트워크 규칙을 삽입하고 제거할 책임이 있습니다. 이에 반해 모니터링 사용자 공간 애플리케이션은 모니터링 맵에서 규칙을 삽입하고 제거할 뿐만 아니라 하드웨어 파이프라인에서 이 맵에 저장된 실시간 통계 및 유용한 정보를 수집합니다. 모니터링 애플리케이션은 소켓당 처리된 초당 패킷, 수신/전송된 패킷 수, 패킷 크기와 같은 Agilio CX SmartNIC에 의해 처리된 네트워크 트래픽의 실시간 지표를 제공합니다. 이 정보는 네트워크 성능에 대한 통찰력을 제공하는 네트워크 관리자와 사용자 모두에게 매우 중요합니다.

C. 슬라이스 정의: 최종 사용자 및 슬라이스 관리 개요

이 연구는 트랜스포트 네트워크 슬라이스의 데이터 플레인을 제어하고 프로그래밍하기 위한 eBPF-XDP 기반 솔루션을 탐구하지만, 이는 네트워크 슬라이스를 모니터링, 관리 및 조정할 수 있는 기능을 갖춘 더 큰 아키텍처 설계의 일부라는 점을 언급하는 것이 중요합니다【35】. 슬라이스의 관리 및 수명 주기를 지원하는 데 필요한 구성 요소에 대한 심층적인 정보는【38】에서 찾을 수 있습니다. 또한【39】의 연구에서는 유럽 6G Brains 프로젝트에서 Open Network Automation Platform(ONAP)과 이 슬라이스 관리자 간의 통합 프로세스를 제시합니다. 이 연구는 서비스 인스턴스 생성 및 슬라이스 인스턴스화에 대한 워크플로를 종합적으로 다루며 최종 사용자에게 실용적인 관점을 제공합니다. 더 깊은 이해를 위해 6G Brains 프로젝트의 Deliverable D5.2【40】를 참조하는 것이 좋습니다. 단순성을 위해 여기서는 해당 단계를 생략하고 대신 데이터 플레인 프로그래밍 가능성(DPP) 애플리케이션에서 수신된 메시지를 직접 설명합니다. 예를 들어, 목록 1은 DPP에서 수신된 슬라이스 정의 메시지를 보여줍니다(그림 2 참조). 메시지는 "자원" 섹션에 개략적으로 설명된 일련의 네트워크 속성으로 정의되는 특정 서비스를 자세히 설명합니다. 이 메시지에는 서비스 우선순위, 최대 허용 대역폭(MAB) 및 최소 보장 대역폭(MGB)과 같은 슬라이스 구성 속성도 포함되어 있습니다. 이 메시지를 통해 DPP는 필요한 패킷 메타데이터 구조(목록 2 참조) 및 해당 해시 식별자를 처리하고 생성할 수 있는 충분한 정보를 얻을 수 있습니다. 마찬가지로 "우선순위" 속성은 eBPF 프로그램에서 이 서비스에 대한 대상 rx 큐를 나타내는 데 사용됩니다(알고리즘 1, 줄 10 참조).

```
{
  "Resources": [{
        "resourceId": "F8A4C949",
        "encapsulationID1": "00000445",
        "encapsulationType1": "vxlan",
        "srcIP": "146.191.50.26",
        "dstPort": "4789",
},{
        "resourceId": "B6E6B2B3",
        "encapsulationID2": "8894D0D4",
        "encapsulationType2": "gtp",
        "srcIP": "10.100.0.19",
        "dstPort": "2152",
}],
  "Intent": {
    "flowAgentName": "XDP-based"
    "actionType": "INSERT",
    "actionName": "CREATE_SLICE",
    "slice_id": "03E8",
    "priority": "1",
    "MAB" : "12000000",
    "MGB" : "10000000"
 },
  "Params": [{
    "paramName":  "interfaceName",
    "paramValue": "eth0"
    }]
}
```

<Listing 1: 의도 기반 메시지 (간단한 버전)>

D. 슬라이스 정의: 제안된 모델, 정책 및 논리적 구현

이 연구에서는 인터넷 엔지니어링 태스크 포스(IETF)가 제시한 트랜스포트 슬라이스 정의를 채택합니다【4】. 이 정의에 따르면, 기반 트랜스포트 네트워크는 슬라이스 내 모든 트래픽 또는 특정 플로우에 대해 일부 또는 모든 서비스 수준 목표(SLO)를 충족하는 트랜스포트 전송을 제공하기 위해 컨트롤러를 통해 네트워크 장치를 동적으로 구성할 수 있어야 합니다. 이를 달성하기 위해 DPP 네트워크 애플리케이션(Figure 2)을 활용합니다. 이 애플리케이션을 사용하면 맞춤형 eBPF 알고리즘과 제어 맵을 통해 기본 smartNIC을 프로그래밍하여 네트워크 장치를 주문형으로 구성할 수 있습니다. 이 접근 방식은 eBPF 시스템 호출을 통해 사용자 공간, 커널 및 하드웨어 간의 제어 통신을 가능하게 합니다. Linux 소스 코드는 /usr/include/linux/bpf.h에서 enum bpf map type을 사용하여 eBPF 맵을 정의합니다. 미리 계산된 플로우 해시 식별자와 연결되어야 하는 특정 큐를 포함한 각 네트워크 플로우의 슬라이스 정의에 대한 정보를 저장하기 위해 BPF MAP TYPE HASH 맵 유형을 사용합니다. 이 데이터 구조의 각 항목은 이 연구에서 "제어 규칙"으로 언급됩니다. 이를 통해 eBPF 맵이 네트워크 트래픽의 특정 특성을 처리하도록 최적화되어 네트워크 플로우의 효율적인 제어 및 관리를 가능하게 합니다.

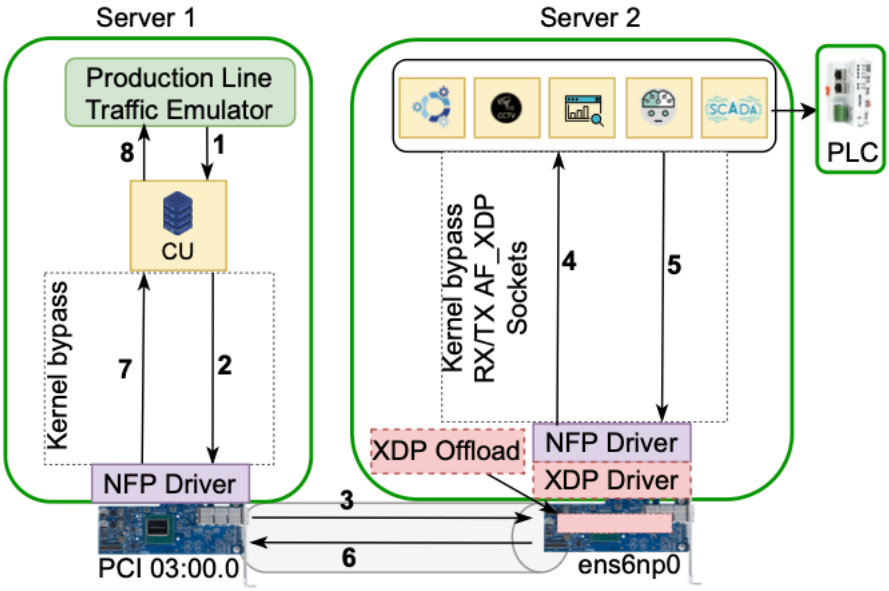
```
struct pkt_meta {
        __u16 out_macsrc[3];
        __u16 out_macdst[3];
        __u16 in_macsrc[3];
        __u16 in_macdst[3];
        __u16 out_ethproto;
        __u16 in_ethproto;
        __be32 out_ip4src;
        __be32 out_ip4dst;
        __be32 in_ip4src;
        __be32 in_ip4dst;
        __u8 out_ip4proto;
        __u8 in_ip4proto;
        union {


                __u32 out_ports;
                __u16 out_port16[2];
        };
        union {
                __u32 in_ports;
                __u16 in_port16[2];
        };
        __u32 vni;
        __be32 teid;
        __u32 ethercatid
};
```

<Listing 2: 패킷 메타데이터 구조>

```
1: struct pkt_meta;
2: procedure XDP_PROG(pkt)
3:     pkt_meta ← parse_headers(pkt);
4:     hash ← calculate_hash(pkt_meta);
5:     control_entry ← control_map.lookup(hash);
6:     if control_entry != NULL then
7:         if control_entry.action == 0 then
8:             return XDP_DROP
9:         if control_entry.action == 1 then
10:            pkt.rx_queue ← control_entry.queue
11:    else
12:        return XDP_PASS
13:    monitor_entry ← monitor_map.lookup(hash);
14:    if monitor_entry != NULL then
15:        monitor.update(hash,pkt_meta,1)
16:    return XDP_PASS;
```

<Algorithm 1 6G 네트워크를 위한 eBPF-XDP 패킷 제어 알고리즘>

알고리즘 1은 C 언어를 사용하여 Agilio CX SmartNIC 내부에 구현된 XDP 오프로드 파이프라인 로직을 보여줍니다. 이 알고리즘은 네트워크 세그먼트를 가속화하는 동시에 트래픽 분류, 필터링 및 제어를 가능하게 합니다. 이 알고리즘을 통해 네트워크 카드와 사용자 공간 간의 트래픽 가속을 보장하기 위해 서로 다른 XSK 소켓 간에 네트워크 플로우를 분산하는 데 사용되는 고유한 해시를 생성하는 데 필요한 메타데이터를 추출하여 두 번 캡슐화된 프리-6G 네트워크 트래픽을 분해할 수 있습니다. SmartNIC 관점에서 패킷이 물리적 인터페이스에 도착하면 오프로드된 eBPF 프로그램이 패킷을 캡처하고 패킷 헤더 추출을 시작합니다(3번째 줄). 이 루틴은 들어오는 패킷의 모든 비트를 구문 분석하고 분석하여 목록 2에 나와 있는 pkt 메타 구조를 완성하는 데 필요한 정보를 추출합니다. 이 데이터 구조에는 내부 및 외부 MAC, IP 주소, 포트, 링크 프로토콜, 전송 프로토콜 및 VXLAN 및 GTP 식별자를 포함하여 프리-6G 네트워크 플로우를 식별하는 데 필요한 정보가 포함되어 있습니다. 이 데이터는 32비트 길이의 해시(4번째 줄)를 계산하는 데 사용되며, 이는 들어오는 네트워크 패킷을 명확하고 개별적으로 식별합니다. 이 해시는 제어 맵에 일치하는 항목이 있는지 확인하는 데 사용됩니다(5번째 및 6번째 줄). 그렇다면 규칙이 확인되고 액션 값이 얻어집니다. 작업 값이 0이면 패킷이 삭제됩니다(7, 8번째 줄). 그러나 동작 값이 1이면 패킷이 사용자 공간으로 전송되기 위해 XSK 소켓으로 전송됨을 의미합니다(9번째 줄). 이를 위해 제어 규칙에서 출력 큐/소켓의 값을 얻고 패킷 구성 메타데이터가 이 값으로 업데이트됩니다(10번째 줄). 해시 값과 일치하는 제어 규칙이 없는 경우 패킷은 기본 큐/소켓(XDP PASS)을 사용하여 사용자 공간으로 전송되며 이는 0입니다(12번째 줄). 패킷이 삭제되지 않은 경우 모니터링 맵을 확인하여 들어오는 패킷의 해시와 일치하는 규칙이 있는지 확인합니다(13번째 줄). 들어오는 패킷의 해시와 일치하는 규칙이 있는 경우(14번째 줄), 현재 처리된 패킷 정보에 추가하고 패킷 카운터를 1씩 늘려 규칙을 업데이트합니다(15번째 줄). 마지막으로 패킷은 5번째 줄에서 이전에 선택한 출력 큐를 사용하여 사용자 공간으로 전송됩니다(16번째 줄).



<Fig 3: 성능 평가 테스트베드 구현>

VI. 실증적 검증

A. 실험 설정

이 섹션에서는 그림 3에 나와 있는 것처럼 제안된 프레임워크의 실증적 검증을 수행하기 위해 사용된 실험 설정을 설명합니다. 이 실험 환경의 개발을 위해 CU를 할당하는 하나와 UPF 및 수직 서비스를 할당하는 두 번째 물리적 서버를 사용하여 클라우드 서비스 제공업체 아키텍처가 복제되었습니다. 두 개의 별도 물리적 서버는 모두 동일한 사양을 가지고 있습니다. CyberServe XE5-212S v4, X10DRi-T, Dual Intel 10GbE LAN, 2TB SATA 7200RPM 및 Intel Solid-State Drive DC P3600 시리즈 1.2TB. 또한 서버 1 및 2는 Agilio CX 2x25GbE SmartNIC을 보유하고 있습니다. 소프트웨어 사양과 관련하여 두 서버 모두 Linux 커널 5.11.0-40-일반이 포함된 운영 체제 Ubuntu 21.04(hirsute)를 가지고 있습니다. 또한 AF XDP 소켓 지원이 포함된 NFP 드라이버의 맞춤형 버전이 두 서버 모두에서 실행되고 있습니다. 서버 1에는 Linux 컨테이너에서 개발되고 OpenAirInterface를 사용하는 CU가 있습니다【41】. 또한 DPDK 21.02.0 및 Pktgen 21.03.1을 사용하는 제품군 트래

픽 에뮬레이터도 포함되어 있습니다. 서버 2에는 SmartNIC에 인스턴스화된 맞춤형 XDP 오프로드 펌웨어, 커널을 우회하는 AF XDP 소켓 및 Linux 컨테이너를 사용하여 에뮬레이션된 다양한 수직 서비스가 있습니다. XDP 펌웨어는 6G 네트워크에 대한 지원 및 제어 및 실시간 모니터링 기능을 갖춘 새로운 파이프라인을 구현합니다. 두 서버는 Avago 25GBase-SR SFP28을 사용하여 연결됩니다.

1. 서버 1에는 센서 역할을 하고 물리적 장치에서 수직 서비스로 비디오를 전송하여 네트워크 트래픽을 생성하는 제품군 트래픽 에뮬레이터라는 도구가 있습니다. 송신기를 에뮬레이션하기 위해 CU 내부에 배포된 Pktgen을 사용합니다. 이 트래픽은 이중 캡슐화를 사용하여 에뮬레이트되며 실시간 프로토콜(RTP) 비디오 프레임을 전송합니다.

2. 서버 1에서 CU에서 생성된 네트워크 트래픽은 전송을 위해 Linux 커널을 우회하여 SmartNIC으로 직접 전송됩니다. 이 통신이 가능하도록 DPDK가 사용되었습니다.

3. 서버 1에서 SmartNIC은 03:00.0 PCI 슬롯을 통해 CU에서 트래픽을 수신하고 물리적 인터페이스를 통해 서버 2로 전달합니다. 최적의 조건에서는 최대 25Gbps의 속도로 트래픽을 전송할 수 있습니다.

4. 서버 1의 PCI 슬롯을 통해 전송된 후 트래픽은 서버 2의 ens6np0 인터페이스를 통해 수신됩니다. 그런 다음 SmartNIC은 오프로드된 파이프라인을 사용하여 트래픽을 처리하고, 패킷 메타데이터를 제어 맵에 저장된 규칙과 대조하여 적절한 조치를 결정합니다. 또한 모니터링 하드웨어 맵은 나중에 성능 분석에 사용되는 들어오는 트래픽에 대한 정보로 업데이트됩니다. 트래픽이 규칙과 일치하지 않으면 삭제됩니다. 그러나 규칙이 일치하면 트래픽은 AF XDP 소켓을 사용하여 수직 서비스로 전달되어 성능을 향상시키기 위해 Linux 커널을 우회합니다. 이 동작은 업스트림된 NFP 드라이버에서 지원되지 않으며 이 연구의 혁신입니다.

5. 그런 다음 네트워크 트래픽은 마침내 수직 서비스에 의해 수신되며, 목표 애플리케이션이 트래픽을 적절하게 처리합니다. 또한 수직 서비스는 SmartNIC 하드웨어 모니터링 맵에서 메트릭을 수집하고 필요한 경우 제어 규칙을 업데이트합니다.

6. 결정을 내리고 네트워크 플로우가 수직 서비스에 의해 패킷의 페이로드에 작업 매개변수를 도입하여 수정되면 이 네트워크 플로우는 동일한 AF XDP 소켓을 사용하고 Linux 커널을 우회하여 SmartNIC으로 다시 전송됩니다.

7. 네트워크 플로우는 ens6np0 물리적 인터페이스를 사용하여 서버 1로 전달됩니다.

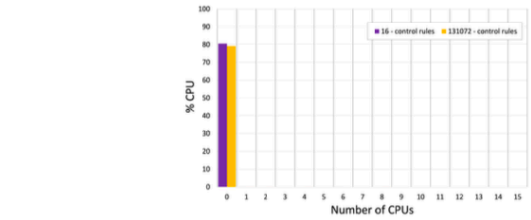
8. 서버 2에 할당된 SmartNIC은 네트워크 트래픽을 CU로 보냅니다.

9. 마지막으로 수직 서비스에 의해 내려진 결정을 포함하는 네트워크 플로우는 제품군 에뮬레이터에 의해 수신되어 관련된 작업을 수행합니다.

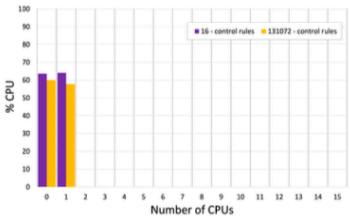
B. 결과

이 섹션에서는 이 논문에서 제안한 프리-6G 네트워크에 대한 가속화된 데이터 경로를 실증적으로 검증합니다. 이 연구의 목표는 미래 6G 네트워크와 인더스트리 4.0 사용 사례와 관련된 까다로운 수직 서비스가 필요한 품질(QoS)과 성능 요구 사항을 충족하는지를 보여주는 것입니다. 이는 실시간 통신이 안전하고 최적의 운영을 위해 중요한 매우 혼잡한 시나리오에서도 높은 신뢰성과 저지연 통신을 제공함으로써 달성됩니다. 이 통신의 성능을 극대화하는 것은 공장의 다양한 제품 라인 간의 통신을 격리하는 데 중요하며, 이를 위해 이 논문에서는 물리적 서버와 제품 라인 간의 통신을 보장하기 위해 격리된 AF XDP 소켓을 제안합니다. 각 소켓에는 시스템의 최대 성능을 얻기 위해 전용 CPU 리소스가 있습니다. 또한, irq affinity가 설정되어 있어 하나의 CPU가 특정 시스템 인터럽트 처리를 담당하기 때문에 CPU 수는 시스템에 예약된 네트워크 수신 큐 수와 비례합니다. 제안된 솔루션의 성능, 신뢰성 및 지연 시간을 검증하기 위해 다양한 차트를 작성하여 이 논문에 제시했습니다. 이 그래프에는 제어 규칙 수, 패킷 크기 및 소켓이 수정될 때 CPU 사용률과 패킷 손실에 대한 통계가 제시되어 있습니다. 또한 패킷 크기를 기준으로 왕복 시간(RTT) 지연 시간도 나타납니다. 패킷 크기에 따라 최대 25Gbps 및 초당 2000만 개의 패킷을 전송하는 실험이 실행되었습니다. 섹션 IV에서 제시한 다양한 사용 사례, 특히 고급 네트워크 슬라이싱의 요구 사항을 테스트하기 위해 패킷 크기는 표 I에서 볼 수 있듯이 최소 6G 이중 캡슐화 패킷에 필

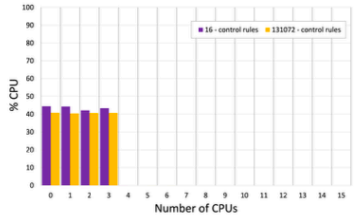
요한 132바이트부터 최대 1500바이트까지입니다. 또한 최대 16개의 소켓을 지원하므로 프레임워크에서 사용하는 최대 CPU 수는 16이며 5가지 다른 실행으로 나뉩니다: 1, 2, 4, 8 및 16개의 소켓. 이 실험은 5회 수행되었으며, 얻은 값은 이 5회의 실행 평균치입니다. 이러한 소켓의 최적 사용은 서로 다른 SLA를 가진 특정 네트워크 서비스를 우선시할 수 있지만, 이 연구는 주로 제안된 솔루션의 확장성을 검증하는 데 중점을 둡니다. 따라서 공정한 검증을 보장하기 위해 모든 전송 서비스 흐름은 사용 가능한 큐/소켓에 고르게 분배됩니다. 더욱이, 기본 아키텍처와 일치시키고 시스템에 주입된 네트워크 트래픽 내에서 일치시키기 위해 제어 eBPF 맵은 특정 제어 규칙으로 미리 채워졌습니다.



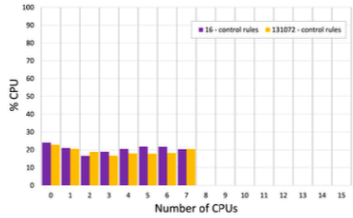
(a) 1: %CPU Usage vs Control Rules with 1 Sockets.



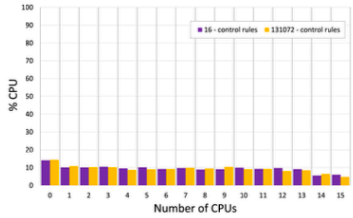
(b) 2: %CPU Usage vs Control Rules with 2 Sockets.



(c) 3: %CPU Usage vs Control Rules with 4 Sockets.



(d) 4: %CPU Usage vs Control Rules with 8 Sockets.



(e) 5: %CPU Usage vs Control Rules with 16 Sockets.

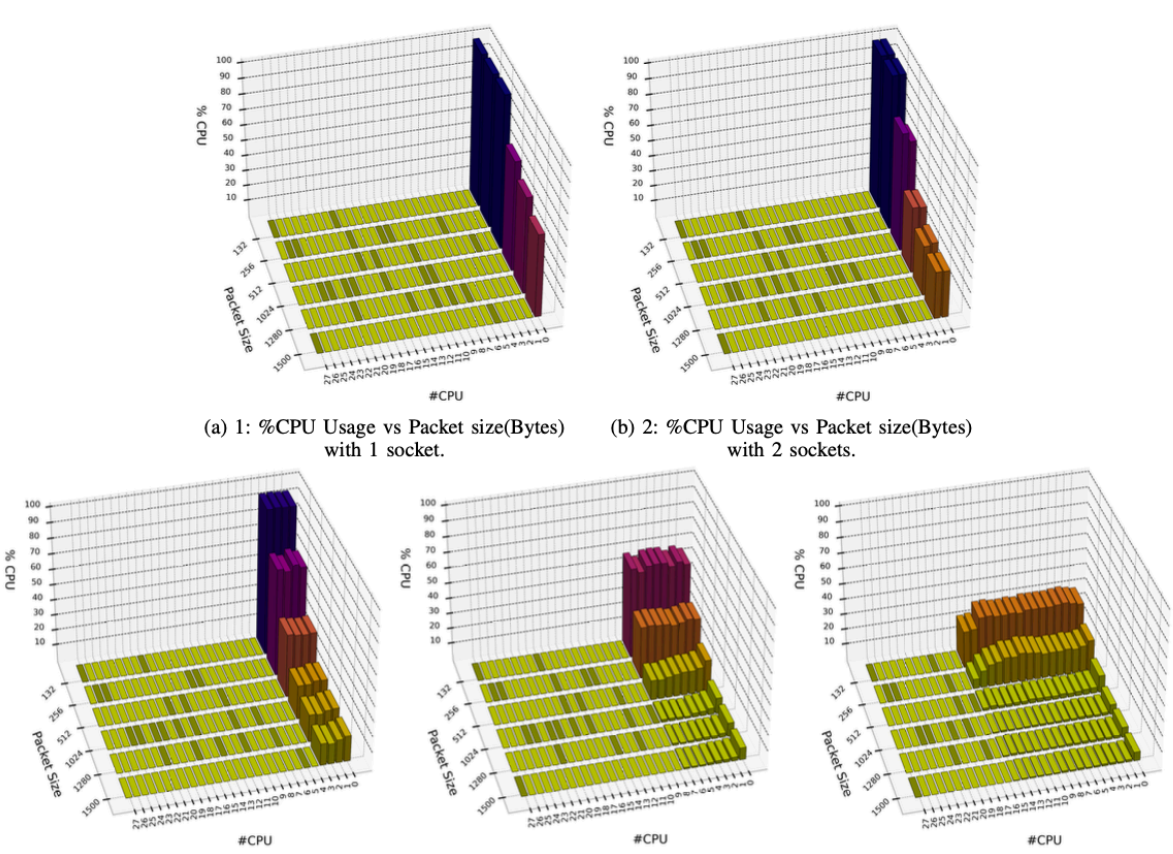
<FIG 4 : CPU 사용률, CPU 수 및 소켓 수에 따른 제어 규칙 수 간의 관계는 다음과 같습니다. 각 CPU가 특정 시스템 인터럽트 처리를 전담하도록 irq affinity가 설정되어 있기 때문에 시스템에 예약된 네트워크 수신 큐 수와 CPU 수는 직접적으로 비례합니다.>.

1) CPU 부하: 그림 4는 최대 16개의 CPU가 동시에 실행되고 SmartNIC 오프로드 맵에 최대 131072개의 규칙이 삽입될 때 사용된 CPU 비율을 보여줍니다. (a), (b), (c), (d), (e)는 각각 1, 2, 4, 8, 16개의 소켓을 사용하여 수행된 다른 실험을 나타냅니다. 수행된 각 실험당 실행 횟수는 16 * 2n이며 n={0, .., 13}입니다. 각 CPU는 특정 시스템 인터럽트 및 특정 CPU를 처리하는 전담 책임이 있습니다. 이 실험을 위해 132~1500바이트의 패킷 크기가 사용되었으며 제시된 결과는 제어 규칙을 사용하여 각 소켓당 균등하게 분배된 모든 트래픽의 평균입니다. 그림 4의 그래프 (a)는 소켓이 하나뿐이기 때문에 모든 네트워크 트래픽이 동일한 CPU에 의해 처리되는 기본 시나리오로 간주될 수 있습니다. 이 CPU는 평균 70%의 부하로 실행 중이며, 그래프에서 볼 수 있듯이 규칙 수가 증가할 때 CPU 부하는 증가하지 않습니다. 그래프 (b)는 두 개의 소켓이 동시에 네트워크 트래픽을 전송하는 시나리오를 나타내며, 그래프에서 볼 수 있듯이 CPU 부하는 평균 50%로 감소했습니다. 마찬가지로 (c), (d) 및 (e)에서는 작동하는 소켓 수가 증가할 때 CPU 부하가 감소합니다. 최고의 시나리오에서 16개의 소켓이 트래픽을 처리할 때 CPU 부하는 10% 미만입니다. 처리된 각 흐름마다 제어 맵을 확인하여 들어오는 네트워크 패킷과 일치하는 규칙이 있는지 확인해야 합니다. 이러한 맥락에서 SmartNIC 하드웨어 맵에 131072개의 제어 규칙이 삽입되면 CPU 부하는 일정하게 유지되어 시스템 부하 측면에서 시스템의 확장성을 입증합니다. 제안된 솔루션 덕분에 최대 25Gbps의 네트워크 트래픽이 처리, 제어 및 모니터링되는 동안 유휴 CPU 부하의 90% 이상이 다른 작업에 사용될 수 있습니다.

그림 5는 패킷 크기가 132바이트에서 최대 1500바이트로 변경될 때 사용된 CPU %를 나타냅니다. 이전 그래프와 마찬가지로 소켓 수는 n={0,..,13}인 크기 16 * 2의 범위입니다. 그래프 (a)는 하나의 소켓만 트래픽을 처리하는 기본 시나리오로 간주될 수 있습니다. 즉, 기본 시나리오는 여러 번 캡슐화된 네트워크 트래픽에 대한 RSS 해시 식별자를 생성할 수 없는 NIC의 동작을 모방합니다. 처리된 패킷의 크기가 512바이트 미만인 경우 시스템에서 처리된 각 패킷당 생성된 소프트웨어 인터럽트 수로 인해 CPU 부하는 거의 100%인 것으로 보입니다. 그러나 이 값은 패킷 크기가 1500바이트로 증가하면 최대 35%까지 감소합니다. 소켓 수가 증가하면 그래프 (b), (c), (d), (e)를 참조하십시오. CPU 부하의 백분율이 감소하여

시스템을 더 유휴 상태로 만들 수 있습니다. 16개의 CPU가 네트워크 트래픽을 처리하는 시나리오에서 132바이트의 패킷 크기로 CPU 사용률은 약 25%이지만 256바이트 패킷 크기로 감소하고 패킷 크기가 512바이트보다 크면 5% 미만으로 감소합니다. 처리된 패킷의 크기가 작고 시스템 인터럽트 수가 매우 많을 때도 제안된 eBPF/XDP 기반 프레임워크의 확장성을 입증합니다.

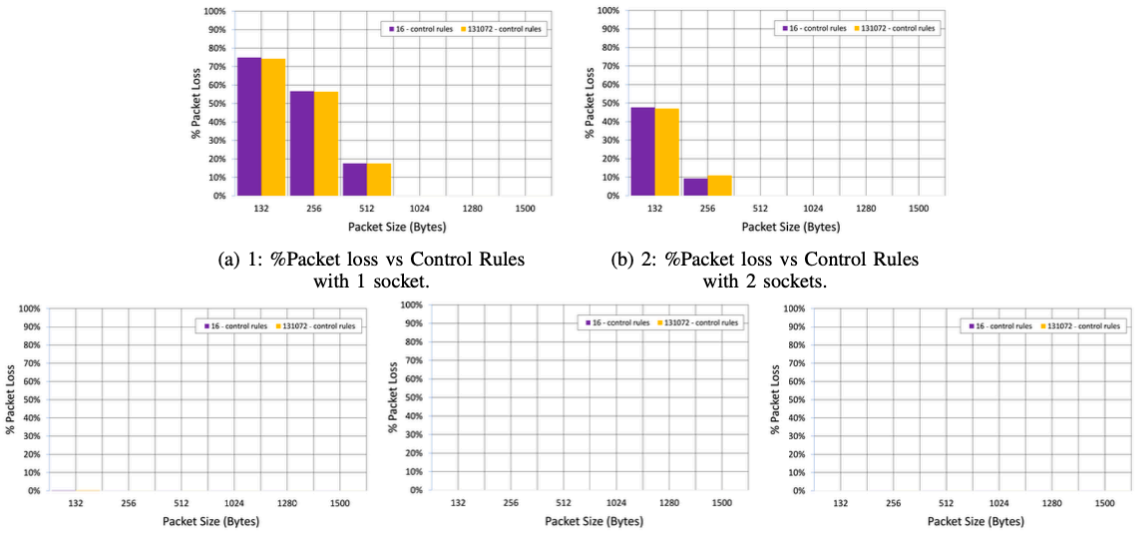
2) 패킷 손실: 다양한 시나리오에서 패킷 손실률을 통해 테스트된 시스템의 신뢰성을 입증할 수 있습니다. 그림 6은 패킷 크기가 변경되고 하드웨어 맵에 삽입된 제어 규칙 수가 변경되었을 때 패킷 손실률을 보여줍니다. 이 검증을 도식화하기 위해 수행된 실험은 이전 실험과 동일하지만, 이번에는 패킷 손실에 중점을 두고 있습니다. 그래프 (a)는 하나의 소켓만 작동하는 기본 시나리오를 보여줍니다. 이 경우 패킷 크기가 132바이트일 때 패킷 손실률은 약 70%입니다. 이 실험에서는 패킷 크기가 1024, 1280 및 1500바이트일 때 거의 0%의 패킷 손실이 발생합니다. 그래프 (b)에서 두 개의 소켓으로 트래픽을 처리할 때, 패킷 크기가 132바이트일 때 솔루션은 여전히 패킷의 약 40%를 잃지만 패킷 크기가 512바이트일 때 거의 0%로 감소합니다. SmartNIC이 처리하는 네트워크 트래픽은 다양한 소켓 간에 고르게 분산되어 있습니다. 그래프 (c), (d) 및 (e)는 프레임워크에 생성된 소켓 수가 4개 이상일 때, 패킷 크기가 132바이트이고 제어 규칙 수가 131072인 최악의 시나리오에서도 패킷 손실이 거의 0%이기 때문에 신뢰성이 거의 100%에 가까워지는 방법을 보여줍니다. 이는 하드웨어 맵에 대한 액세스 시간이 일정하며 제어 규칙 수가 증가할 때 시스템의 신뢰성을 입증한다는 것을 의미합니다.



(a) 1: %CPU Usage vs Packet size(Bytes) with 1 socket.

(b) 2: %CPU Usage vs Packet size(Bytes) with 2 sockets.

<Fig 5 : CPU 사용률, CPU 수 및 소켓 수에 따른 패킷 크기(바이트 단위)의 관계는 다음과 같습니다. 각 CPU가 특정 시스템 인터럽트 처리를 담당하도록 irq affinity가 설정되어 있기 때문에 시스템에 예약된 네트워크 수신 큐 수와 CPU 수는 직접적으로 비례합니다.>



(a) 1: %Packet loss vs Control Rules with 1 socket.

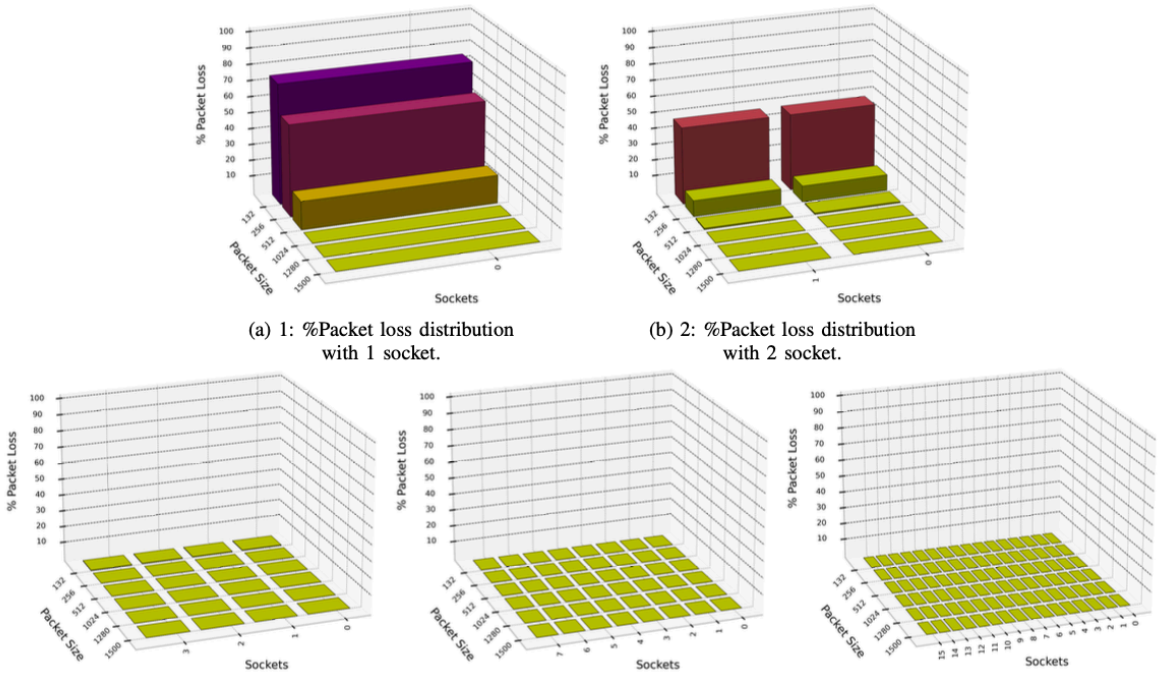
(b) 2: %Packet loss vs Control Rules with 2 sockets.

(c) 3: %Packet loss vs Control Rules with 4 sockets.

(d) 4: %Packet loss vs Control Rules with 8 sockets.

(e) 5: %Packet loss vs Control Rules with 16 sockets.

<Fig6 : 다른 소켓 수를 사용했을 때 % 패킷 손실, 바이트 단위의 패킷 크기 및 제어 규칙 수 간의 관계>



(a) 1: %Packet loss distribution with 1 socket.

(b) 2: %Packet loss distribution with 2 socket.

(c) 3: %Packet loss distribution with 4 socket.

(d) 4: %Packet loss distribution with 8 socket.

(e) 5: %Packet loss distribution with 16 socket.

<Fig7 : % 패킷 손실, 바이트 단위 패킷 크기, 소켓 수 간의 관계.>

이러한 시나리오에서 CPU가 균등하게 분배되어 서로 다른 소켓에 연결되고 이러한 소켓이 서로 완전히 격리되어 있으므로 각 소켓의 성능을 연구하는 것이 흥미롭습니다. 이 경우 트래픽이 서로 다른 소켓 간에 균등하게 분배되었기 때문에 이론적으로 각 소켓의 동작은 동일해야 합니다. 전송된 다양한 네트워크 플로우의 분배는 하드웨어 카드에 삽입된 최대 131,072개의 제어 규칙을 사용하여 수행됩니다. 그림 7에 나타난 것처럼, 각 그래프의 경향은 그림 6에 나타난 그래프와 매우 유사하며, 실제로 동일한 데이터이지만 서로 다른 정보를 보여줍니다. 이 그림에서는 모든 제어 규칙의 평균을 계산했으며 패킷 크기와 소켓 수를 비교하여 보여주는 것에 중점을 두고 있습니다. 다시 그래프 (a)는 패킷 크기가 132바이트일 때 거의 70%의 패킷이 손실되는 기본 시나리오로 간주됩니다. 프레임워크에 사용된 소켓 수가 4개 이상인 경우 이 값은 0이 됩니다.

이 논문의 실증적 검증을 통해 입증된 높은 신뢰성과 고성능은 표 I에 제시된 6G 사용 사례의 까다로운 요구 사항을 충족하기 위한 솔루션의 적합성을 검증합니다. 이 작업은 고급 네트워크 슬라이싱 솔루션을 제공하는 데 중점을 두고 있지만, 생성된 큐 수와 제어 및 모니터링 맵에 삽입된 규칙에 따라 이 솔루션은 이전에 언급한 모든 사용 사례를 성공적으로 배포하는 데 사용할 수 있습니다.

3) 대역폭: 배포된 프레임워크의 확장성 테스트는 그림 8에 표시된 것처럼 패킷 크기와 소켓 수를 수정하는 동안 수신된 초당 패킷 수(PPS)를 계산하여 테스트되었습니다. 시스템의 성능을 평가하기 위해 세 가지 다른 개념을 고려해야 합니다. (a) PPS, 네트워크를 통해 전송되는 초당 패킷 수를 나타냅니다. (b) 네트워크의 최대 데이터 볼륨 용량을 나타내는 대역폭 (c) 네트워크에서 성공적으로 이동하는 데이터 양을 나타내는 처리량. 이 기사에서는 그림 8에 표시된 값은 소켓에서 수신된 PPS(대역폭) 수이고 처리된 PPS(처리량) 수가 아님을 유의하십시오. 소켓당 처리된 총 PPS를 계산하려면 독자가 각 실험을 수행하기 위해 수집한 샘플이 30초 동안 지속된다는 점과 그림 7에 표시된 소켓당 패킷 손실 수를 고려해야 합니다.

총 처리량 또는 대역폭은 패킷 크기에 따라 달라지므로 솔루션의 실제 성능을 나타내지는 않으므로 이러한 그래프는 이 논문에 포함되지 않았습니다. 그러나 이러한 값은 PPS 결과, 소켓당 수신된 대역폭을 기반으로 직접 계산할 수 있습니다.

```

BW(Gbps) = (PPS * PacketSize)/10^9 또는 소켓당 처리된 처리량

TP(Gbps) = ((PPS * PacketSize)/10^9) - PacketLoss

```

이 실험을 위해 패킷 생성기에서 생성된 패킷의 동일한 수가 생성된 다른 소켓으로 전송되므로 각 소켓에서 거의 동일한 양의 네트워크 트래픽이 동시에 처리됩니다. 8 그래프 (a)에서 볼 수 있듯이 전송된 패킷 크기가 132B일 때 제안된 프레임워크에서 수신된 PPS 수는 거의 2000만 개입니다. 그러나 이것이 기본값이자 최악의 시나리오이며 처리되는 동안 이 패킷의 거의 70%가 손실됩니다. 소켓 수가 증가하면 수신되는 트래픽 양은 사실상 동일하지만 패킷 손실 수는 감소합니다. 패킷 크기가 증가하면 소켓당 수신되는 패킷 수가 감소하는데, 이는 이 실험에서 SmartNIC에서 지원하는 최대 대역폭(25Gbps)이 더 적은 패킷으로 달성되기 때문입니다. 그래프 (c)에서 입증된 것처럼 거의 모든 패킷을 손실하지 않고 각 소켓이 처리한 최대 PPS는 약 500만 PPS입니다. 수행된 실험에서 25Gbps의 최대 대역폭으로 소켓 수가 4개 이상일 때 최대 성능이 달성되며 모든 실행된 실험에서 거의 0%의 패킷 손실이 있습니다. 더욱이, 달성된 대역폭 값은 표 I에 표시된 다양한 프리-6G 수직 사용 사례에서 필요한 평균 대역폭을 충족하며 대역폭 측면에서 제안된 솔루션의 타당성을 입증합니다.

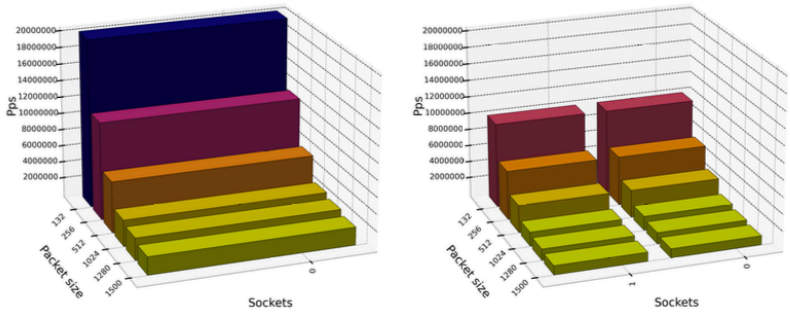
C. 지연 시간

최신 프리-6G 네트워크의 중요한 핵심 성능 지표는 네트워크 효율성, 연결 속도 및 대역폭과 관련된 지연 시간입니다. 네트워크 전송 지연 시간이 줄어들면 최종 사용자의 경험 품질(QoE)에 직접적인 영향을 미칩니다. 그림 9는 그림 3에 나타난 두 서버 간의 다양한 프리-6G 모바일 네트워크 통신에 대한 평균 RTT를 보여줍니다. 이 서버들은 생산 라인과 클라

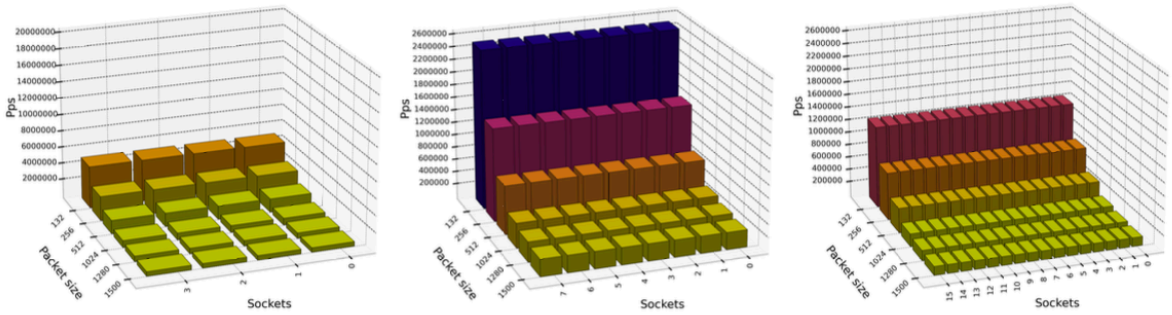
우드 서비스 제공 인프라를 구성합니다. 실험에서는 이전 실행과 동일한 패킷 크기를 사용했으며 최대 처리 속도는 25Gbps입니다. 표시된 값은 패킷 생성기 서버에서 출발하여 광케이블을 통해 네트워크를 통과하고, 오프로드 맵을 사용한 XDP 하드웨어 데이터 경로를 거쳐, AF XDP 드라이버 소켓을 통해 수직 애플리케이션에 도달하는 패킷의 지연 시간을 나타냅니다. 이 과정에는 수직 애플리케이션의 트래픽 처리와 동일한 소켓과 케이블을 통한 원점 복귀도 포함됩니다. 이 연구는 패킷이 AF XDP 소켓에 도달하는 단계까지 집중하지만 보고된 RTT 시간에는 제안된 솔루션의 범위를 벗어나는 사용자 공간(수직 애플리케이션)의 처리까지 포함한 전체 프로세스가 포함됩니다. 그래프는 패킷 크기가 커짐에 따라 RTT가 증가함을 보여주며, 이는 주로 수직 애플리케이션에서 처리하는 데 더 많은 시간이 필요하고 새 패킷을 생성하여 다시 네트워크로 재주입하는 것에 기인합니다. 그러나 하드웨어 및 드라이버 수준에서 가장 까다로운 시나리오는 25Gbps에서 132B 패킷이 전송될 때 발생하며, 이는 총 2000만 개의 패킷이 초당 처리되어야 함을 의미합니다. 이 시나리오에서 RTT는 약 0.13ms입니다. 표 I에 따르면 프리-6G 네트워크에서 이 기사에서 설명한 중요 서비스에 대한 최대 허용 지연 시간은 종단 간 통신의 경우 0.1ms입니다. 제안된 솔루션이 0.1ms에 가까운 왕복 전송을 달성할 수 있다는 점에서 이 프레임워크가 프리-6G 네트워크의 엄격한 지연 시간 KPI를 충족할 수 있음을 알 수 있습니다. 그럼에도 불구하고 Use Case 5와 같이 큰 패킷 크기와 매우 낮은 지연 시간이 필요한 특정 시나리오에서는 어려움이 발생할 수 있습니다. 그림 9는 패킷 크기가 1500바이트인 경우 평균 RTT가 0.55ms로 증가함을 보여줍니다. 이러한 경우에는 더 높은 기능을 갖춘 smarNIC, 트래픽을 분산하기 위한 더 많은 하드웨어 큐, 그리고 사용자 공간 수준에서 네트워크 트래픽을 큐에서 꺼내고 처리하기 위한 서버 CPU가 필요할 것입니다. 또 다른 대안은 이전 연구 기사【7】에서 탐구한 것과 유사하게 로드 밸런서와 같이 다른 서버 간에 트래픽을 분산시키는 것입니다.

VII. 결론

차세대 모바일 네트워크의 까다로운 성능 요구 사항과 새로운 사용 사례에는 고성능 패킷 처리, 제어 및 모니터링 기능이 필요합니다. 이 논문에서는 미래 6G 네트워크를 위한 새로운 초고성능 패킷 처리 및 모니터링 네트워크 프레임워크를 제안했습니다. 제안된 새로운 eBPF/XDP 기반 네트워크 필터링 메커니즘은 멀티 테넌트 흐름을 동적으로 식별하고 구체적인 프리-6G 네트워크 인더스트리 4.0 사용 사례의 요구 사항을 충족하기 위해 처리된 트래픽에 대한 결정을 내릴 수 있습니다. 이와 동시에 이 프레임워크는 메모리에 대한 지속적인 접근 시간을 제공하는 eBPF 하드웨어 맵을 사용하여 실시간 네트워크 메트릭을 보고할 수 있습니다. 제안된 프로그래밍 가능한 네트워크 데이터 플레인은 eBPF, XDP 및 AF XDP와 같은 새로운 기술을 결합하여 Linux 커널을 우회하는 Agilio CX 스마트 네트워크 인터페이스 카드를 사용하여 하드웨어 기반 오프로드 접근 방식을 탐색하여 네트워크 카드와 사용자 애플리케이션 간의 통신을 가속화합니다. 이를 지원하기 위해 SmartNIC을 제어하는 NFP 드라이버가 AF XDP 지원을 제공하도록 수정되었습니다. 이 프레임워크는 현실적인 5G 이상의 아키텍처에서 구현 및 테스트되었으며 제안된 솔루션의 타당성을 입증하는 실험 결과를 얻었습니다. 또한 다양한 시나리오에서 패킷 손실과 CPU 사용률 측면에서 유리한 성능 결과를 통해 AF XDP 소켓 수와 제어 규칙을 수정하여 솔루션이 다양한 프리-6G 사용 사례의 까다로운 KPI를 충족할 수 있는 확장성과 적합성을 명확하게 보여줍니다. 더욱이 현재 연구의 특정 제한 사항을 해결할 계획입니다. 예를 들어, 복잡한 네트워크 토폴로지에서 새로운 과제를 식별하기 위해 연결 지점 간의 지리적 거리와 같은 요인을 조사할 것입니다. 이러한 영역에서 연구를 확장함으로써 미래를 위한 견고하고 효과적인 네트워크 솔루션 개발에 기여할 수 있기를 바랍니다.



(a) 1: Pps vs Packet size(Bytes) vs Sockets. (b) 2: Pps vs Packet size(Bytes) vs Sockets.



(c) 3: Pps vs Packet size(Bytes) vs Sockets. (d) 4: Pps vs Packet size(Bytes) vs Sockets. (e) 5: Pps vs Packet size(Bytes) vs Sockets.

Fig. 8: Relation between packet per second received, packet size in bytes and number of sockets.

(a) 1: 초당 패킷 수(PPS) vs 패킷 크기(바이트) vs 소켓.

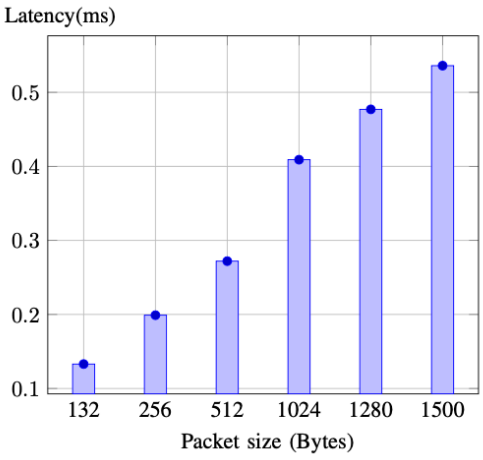
(b) 2: 초당 패킷 수(PPS) vs 패킷 크기(바이트) vs 소켓.

(c) 3: 초당 패킷 수(PPS) vs 패킷 크기(바이트) vs 소켓.

(d) 4: 초당 패킷 수(PPS) vs 패킷 크기(바이트) vs 소켓.

(e) 5: 초당 패킷 수(PPS) vs 패킷 크기(바이트) vs 소켓.

<Fig 8: 초당 수신된 패킷 수, 바이트 단위 패킷 크기, 소켓 수 간의 관계.>



<Fig 9: 패킷 크기가 변할 때 RTT 지연 시간(밀리초 단위).>

REFERENCES

1. [1] W. Saad, M. Bennis, M. Chen, and A. V. Vasilakos, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 35, no. 1, pp. 186–192, 2021.
2. [2] P.Singh,A.Nayyar,A.Kaur,andU.Ghosh,"Blockchainandfogbased architecture for internet of everything in smart cities," *Future Internet*, vol. 12, no. 4, p. 61, 2020.
3. [3] P. K. Padhi and F. Charrua-Santos, "6g enabled industrial internet of ev- erything: towards a theoretical framework," *Applied System Innovation*, vol. 4, no. 1, p. 11, 2021.
4. [4] R. R. Tantsura, S. Homma, K. Makhijani, L. M. Co, ntreras, and Jeff, "IETF Definition of Transport Slice," 2020. [Online]. Available: https: //datatracker.ietf.org/doc/html/draft- nsdt- teas- transport- slice- definition
5. [5] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5g: Challenges and opportunities," *IEEE Internet Computing*, vol. 23, no. 5, pp. 16–23, 2019.
6. [6] X. Foukas, G. Patounas, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1669–1713, 2020.
7. [7] P. Salva-Garcia, R. Ricart-Sanchez, E. Chirivella-Perez, Q. Wang, and J. M. Alcaraz-Calero, "Xdp-based smartnic hardware performance accel- eration for next-generation networks," *Journal of Network and Systems Management*, 2022.
8. [8] M. Majkowski, "Kernel bypass," *https://blog.cloudflare.com/ kernel-bypass/*, 2015.
9. [9] H.Zhu,*DataPlaneDevelopmentKit(DPDK):ASoftwareOptimization Guide to the User Space-based Network Applications*. CRC Press, 2020.
10. [10] M. Paolino, N. Nikolaev, J. Fanguede, and D. Raho, "Snabbswitch user space virtual switch benchmark and performance optimization for nfv," in *2015 IEEE Conference on Network Function Virtualization and*

    Software Defined Network (NFV-SDN). IEEE, 2015, pp. 86–92.

11. [11] V. Maffione, L. Rizzo, and G. Lettieri, "Flexible virtual machine networking using netmap passthrough," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2016, pp. 1–6.
12. [12] L. Kernel, "Linux packet mmap," *https:// www.kernel.org/ doc/ Documentation/ networking/ packet\ mmap.txt*, accessed: July 15,2023.
13. [13] ntop, "PF RING: High-speed packet capture, filtering and analysis,"

    2023, accessed: July 23,2023. [Online]. Available: https://www.ntop.

    org/products/packet- capture/pf ring/

14. [14] Xilinx, "Ef vi user guide," *https:// china.xilinx.com/ content/*

    dam/ xilinx/ publications/ solarflare/ onload/ openonload/ packages/

    SF- 114063- CD- 10 ef vi User Guide.pdf , accessed: July 15,2023.

15. [15] Kernel, "Af xdp - af xdp socket (xsk)," *https://www.kernel.org/doc/*

    html/latest/networking/af xdp.html, accessed: July 15,2023.

16. [16] A. Xilinx, "X2 series ethernet adapters - xtremescale x2522, x2541,"

    https:// www.xilinx.com/ products/ boards- and- kits/ x2- series.html,

    accessed: July 20,2023.

17. [17] A. Xilinx, "8000 series ethernet adapters - 10/40gbe network adapters,"

    https://www.xilinx.com/products/boards-and-kits/8000-series.html, ac-

    cessed: July 20,2023.

18. [18] A. Xilinx, "Alveo u50 data center accelerator card," *https://www.xilinx.*

    com/products/boards-and-kits/alveo/u50.html, accessed: July 20,2023.

19. [19] A.Xilinx,"Alveou200datacenteracceleratorcard,"*https://www.xilinx. com/ products/ boards- and- kits/ alveo/ u200.html*, accessed: July 20,2023.

20. [20] A. Xilinx, “Alveo u250 data center accelerator card,” *https://www.xilinx. com/ products/ boards- and- kits/ alveo/ u250.html*, accessed: July 20,2023.
21. [21] A. Xilinx, “Alveo u280 data center accelerator card,” *https://www.xilinx. com/ products/ boards- and- kits/ alveo/ u280.html*, accessed: July 20,2023.
22. [22] Intel, “Intel fpga smartnic n6000-pl platform,” *https:// www.intel.*

com/ content/ www/ us/ en/ products/ details/ fpga/ platforms/ smartnic/

n6000-pl-platform.html, accessed: July 20,2023.

23. [23] Silicom, “Silicom fpga smartnic n5010 series,” *https:// www.silicom- usa.*

com/ pr/ fpga- based- cards/ fpga- intel- based/ fpga- intel- stratix- based/

silicom-fpga-smartnic-n5010 series/, accessed: July 20,2023.

24. [24] Intel, “Intel fpga programmable acceleration card n3000,”

https:// www.intel.com/ content/ www/ us/ en/ products/ details/ fpga/

platforms/ pac/ n3000.html, accessed: July 20,2023.

25. [25] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore,

“Netfpga sume: Toward 100 gbps as research commodity,” *IEEE micro*,

vol. 34, no. 5, pp. 32–41, 2014.

26. [26] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford,

C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, “P4: Pro- gramming protocol-independent packet processors,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

27. [27] J.Cavanagh,*DigitaldesignandVerilogHDLfundamentals*. CRCpress, 2017.
28. [28] R. Ricart-Sanchez, A. C. Aleixo, Q. Wang, and J. M. A. Calero, “Hardware-based network slicing for supporting smart grids self-healing over 5g networks,” in *2020 IEEE International Conference on Commu- nications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.
29. [29] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang, “P4-netfpga-based network slicing solution for 5g mec architectures,” in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2019, pp. 1–2.
30. [30] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang, “Netfpga-based firewall solution for 5g multi-tenant architectures,” in *2019 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2019, pp. 132–136.
31. [31] Q. Wang, J. Alcaraz-Calero, R. Ricart-Sanchez, M. B. Weiss, A. Gavras, N. Nikaein, X. Vasilakos, B. Giacomo, G. Pietro, M. Roddy *et al.*, “Enable advanced qos-aware network slicing in 5g networks for slice- based media use cases,” *IEEE transactions on broadcasting*, vol. 65, no. 2, pp. 444–453, 2019.
32. [32] Netronome, “Netronome agilio cx smartnics,” *https://www.netronome. com/products/agilio-cx/*, accessed: July 21,2023.
33. [33] Corigine, “Corigine agilio cx smartnics,” *https://www.corigine.com/ smartnicdetail-31.html*, accessed: July 15,2023.
34. [34] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjo¨land, and F. Tufvesson, “6g wireless systems: Vision, requirements, challenges, insights, and opportunities,” *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, 2021.
35. [35] A. Artemenko, Y. Zhang, U. Wostradowski, J. Cosmas, Q. Wang, J. M. Alcaraz-Calero, E. C. Perez, P. Salva-Garcia, and R. Ricart-Sanchez, “6G BRAINS: D2.1 Definition and Description of the 6G Primary Use Cases and Derivation of User Requirements,” Jun. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5185388

[36] L. Kernel, “Libbpf library,” *https:// www.kernel.org/ doc/ html/ latest/ bpf/ libbpf/ index.html*, accessed: July 11,2023.

[37] L. Kernel, “Xsk library,” *https:// github.com/ torvalds/ linux/ blob/ master/ tools/ lib/ bpf/ xsk.c*, accessed: July 11,2023.

[38] E. Chirivella-Perez, P. Salva-Garcia, I. Sanchez-Navarro, J. M. Alcaraz- Calero, and Q. Wang, “E2e network slice management framework for 5g multi-tenant networks,” *Journal of Communications and Networks*, pp. 1–13, 2023.

[39] J. Fonseca, M. Khadmaoui-Bichouna, B. Mendes, P. Duarte, M. Araujo, D. Corujo, I. Sanchez-Navarro, A. Matencio-Escolar, P. Salva-Garcia, J. M. Alcaraz-Calero, and Q. Wang, “6g brains topology-aware industry- grade network slice management and

orchestration," in *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2023, pp. 341–346.

[40] A. Kazmierowski, J. Kodjabachian, P. Salva-Garcia, A. Matencio- Escolar, and I. Sanchez-Navarro, "6G Brains: D5.2 Preliminary integration for AI-based E2E network slicing control and MANO," Dec. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7468007

[41] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.