An eBPF-XDP hardware- based network slicing architecture for future 6G front to back haul networks

김도현

목차

개요

이 논문은 5G와 6G 네트워크에서 높은 데이터 전송 속도, 신뢰성, 유연한 네트워크 관리의 필요성에 대해 설명합니다.

네트워크 슬라이싱, 프로그래밍 가능한 데이터 플레인을 활용하여 XDP, eBPF, SmartNICs와 같은 기술을 통해 네트워크 성능과 효율성을 극대화하는 방법을 설명합니다..

XDP

- 고성능 패킷 처리를 위해 네트워크 인터페이스 카드에서 수신되는 패킷을 최소한의 지연으로 처리할 수 있는 프레임워크

eBPF

- 리눅스 커널에서 안전하고 효율적인 방식으로 코드를 실행할 수 있는 프로그래밍 모델

SmartNICs

- SmartNIC은 네트워크 인터페이스 카드에 프로세싱 능력을 더해 네트워크 트래픽을 CPU에 부담을 주지 않고 독립적으로 처리할 수 있음

네트워크 슬라이싱

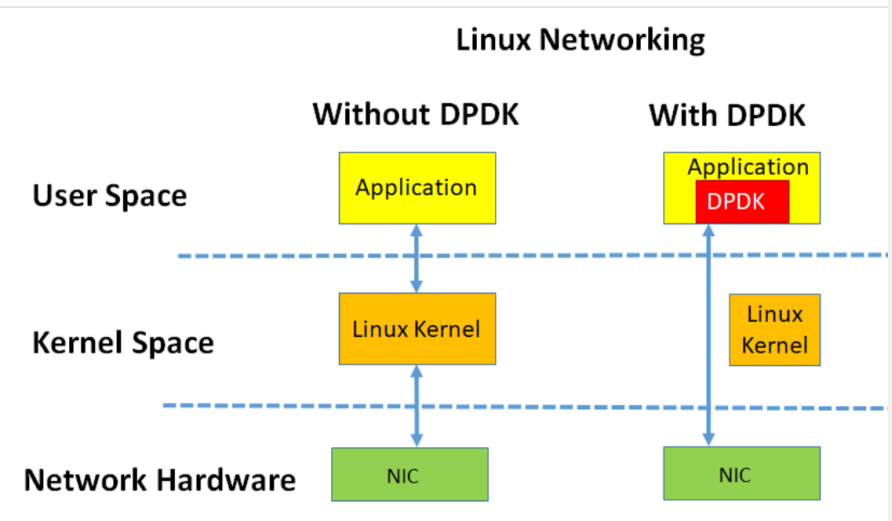
- 하나의 물리적 네트워크를 여러 독립적인 가상 네트워크로 분할시키는 기술

프로그래밍 가능한 데이터 플레인

- 네트워크 패킷을 처리하는 로직을 사용자가 정의하고 변경할 수 있는 아키텍쳐

커널 바이패스 기술이란?

컴퓨터에서 데이터를 처리할 때 운영 체제의 중심부 커널을 우회하는 기술



- 높은 네트워크 성능
- 낮은 지연시간
- CPU 부하 감소

종류: DPDK, AF_XDP, NETMAP, PF_RING, SNABBSWITCH

하드웨어 오프로드 기술이란?

프로그래밍 가능한 하드웨어를 사용하여 처리 속도를 유지하고 필터링 규칙을 증가시켜도 추가 메모리 액세스 시간의 부담을 없애는 기술

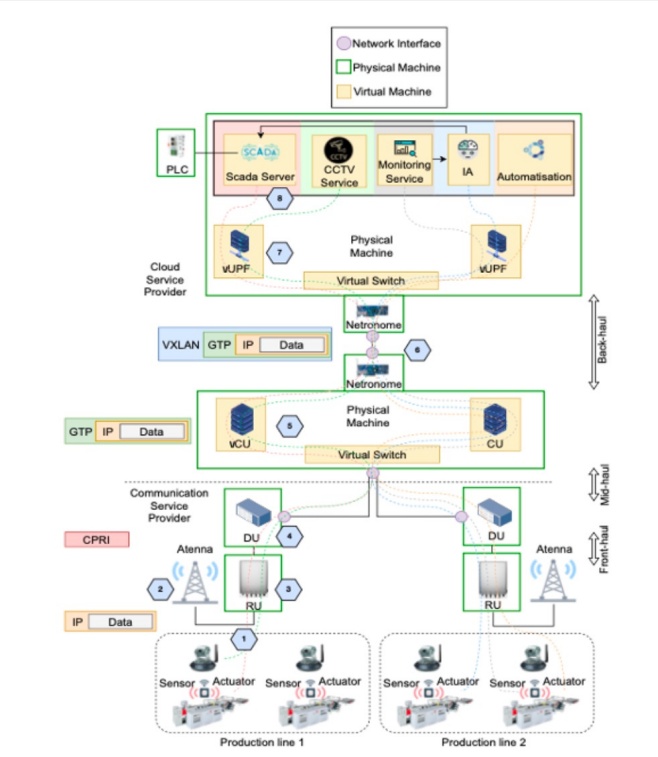
- FPGA SmartNIC : 높은 가격, SDK 및 드라이버의 사유권 문제
- INTEL SmartNIC : 독점적인 코드로 선호X
- NETFPGA 오픈소스 솔루션은 6G KPI를 충족X

해결방안

- AGILO SMARTNIC 사용

➡️ c언어를 이용하여 SmartNIC을 프로그래밍 가능, AF_XDP 지원

멀티 테넌트 pre-6G 아키텍처

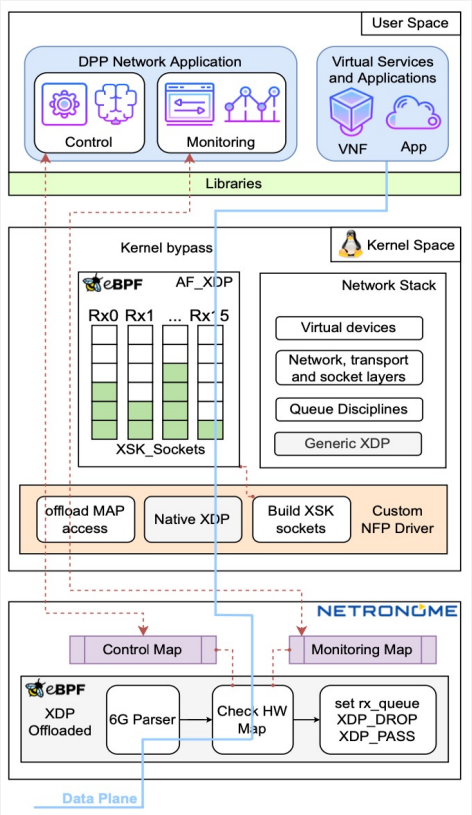


1. CSP(통신 서비스 제공업체)
2. MIMO 안테나
3. RU
4. DU
5. vCU
6. SmartNIC
7. vUPF
8. GTP, VXLAN 사용하여, 특정 산업 또는 응용 분야에 맞춤형 서비스를 제공

RU : Radio Unit
DU : Distributed Unit
CU : Central Unit
vUPF : virtualized User Plane Function

GTP: GPRS Tunneling Protocol
VXLAN : virtual Extensible LAN

제안된 프레임워크 아키텍처



User Space
- DPP 네트워크 애플리케이션
- 가상 서비스 및 애플리케이션
  - 제어 애플리케이션
  - 모니터링 애플리케이션

Agilio CX SmartNIC
eBPF 맵
- 제어맵 (Control Map)
- 모니터링 맵(Monitoring Map)

Linux 커널 공간
- NFP 드라이버 사용

➡️ 일반형은 스택 병목현상 일어남

- 맞춤형 NFP 드라이버

➡️AF XDP와 XSK 소켓을 활용하여 단점 보완

Xsk 소켓 : XDP와 연동하여 고성능 네트워크 패킷을 처리를 가능하게 하는 소켓 인터페이스, 고성능, 유연성, BPF 사용하여 효율적으로 패킷 필터링과 처리함.
DPP : Data Plane Programming

최종 사용자 및 슬라이스 관리


```
{
  "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"
    }]
}
```


DPP 애플리케이션을 통해 수신된 슬라이스 정의 메시지에 집중
필요한 패킷 메타 데이터 구조 및 식별자 처리하고 생성

eBPF-XDP 패킷 제어 알고리즘

```
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;
```

<알고리즘 3줄 요약>

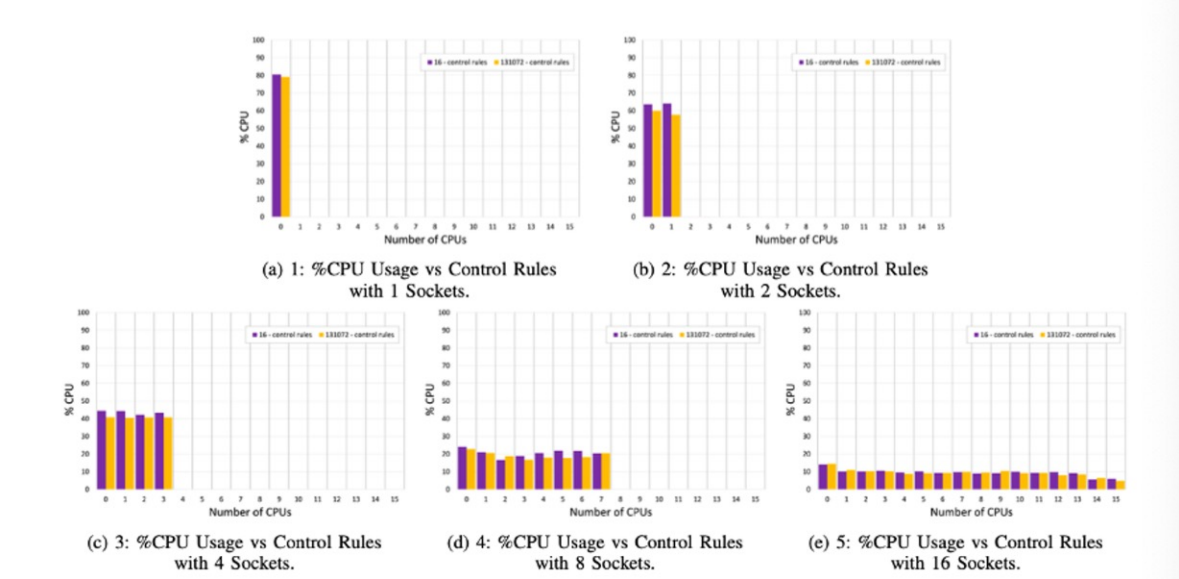
패킷에서 메타 데이터를 추출하고 해시 값 계산

제어 맵에서 드롭, 전송, 통과 결정

패킷 처리 결과를 모니터링 맵에 기록

검증단계

CPU 부하 분석



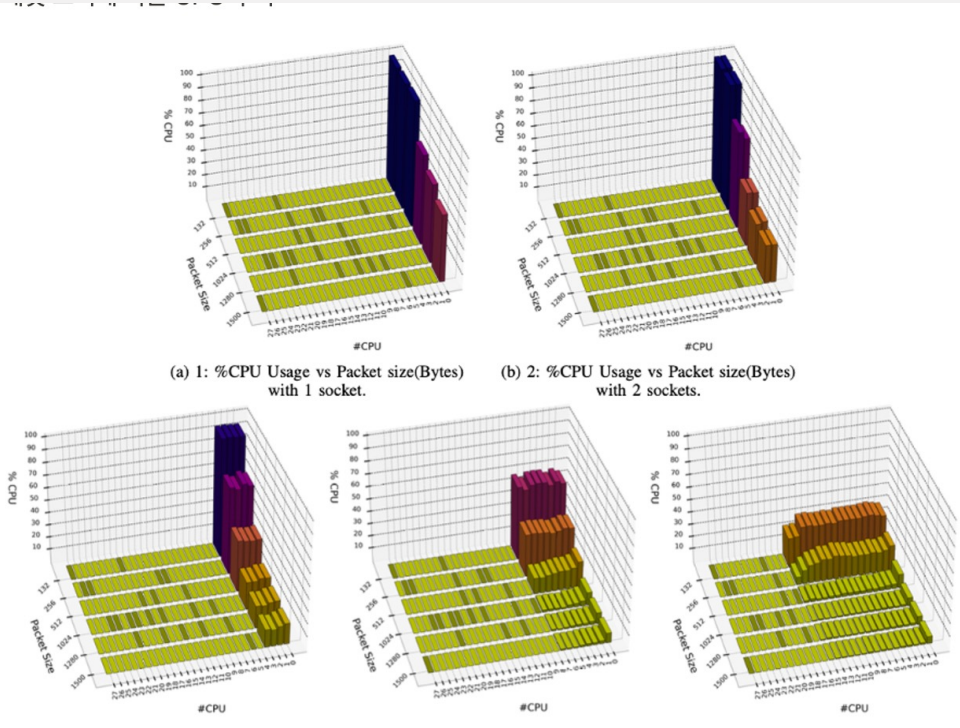
<조건 : 최대 131072개의 제어 규칙이 삽입된 상태>

소켓 수가 증가함에 따라 CPU 부하 ⬇️

제어 규칙 수에 상관없이 CPU 부하 일정하게 유지

시스템의 확장성 입증

패킷 크기에 따른 CPU 부하



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

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

소켓이 1개이고 패킷 크기가 512바이트 이하
- CPU 100% 사용

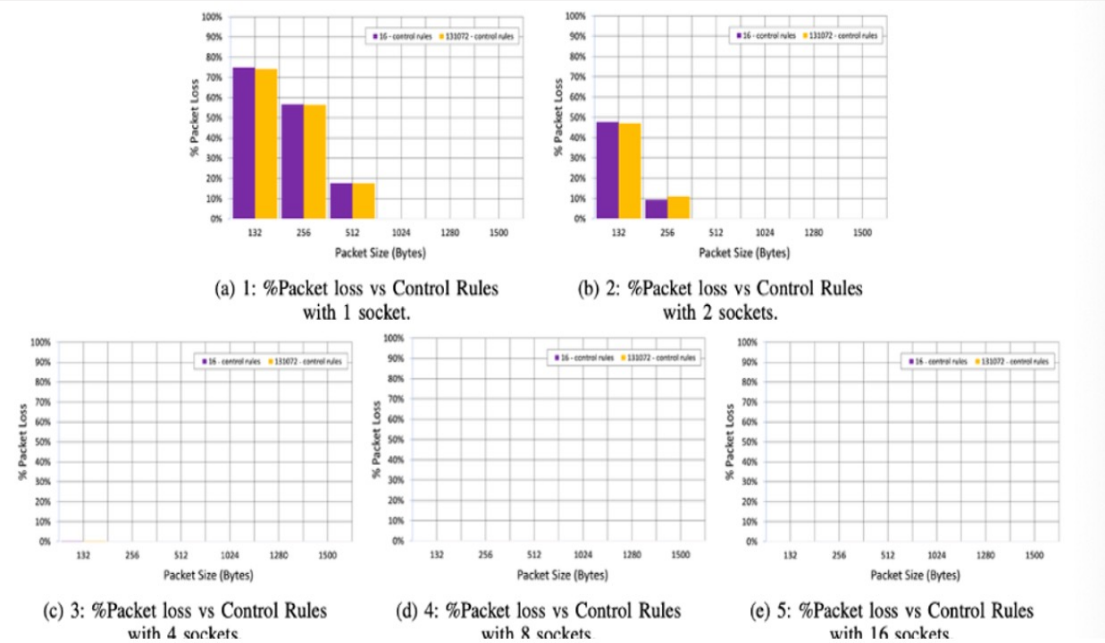
BUT, 패킷 크기가 512 바이트 이하
- CPU 35% 사용

16 개 CPU에서 132 패킷
- CPU 25% 사용

BUT, 512바이트 이상일 시
- CPU 5% 사용

즉, 패킷 크기 ⤴️ CPU 부하 ⬇️
소켓 수 ⤴️ CPU 부하 ⬇️

패킷 손실률



(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

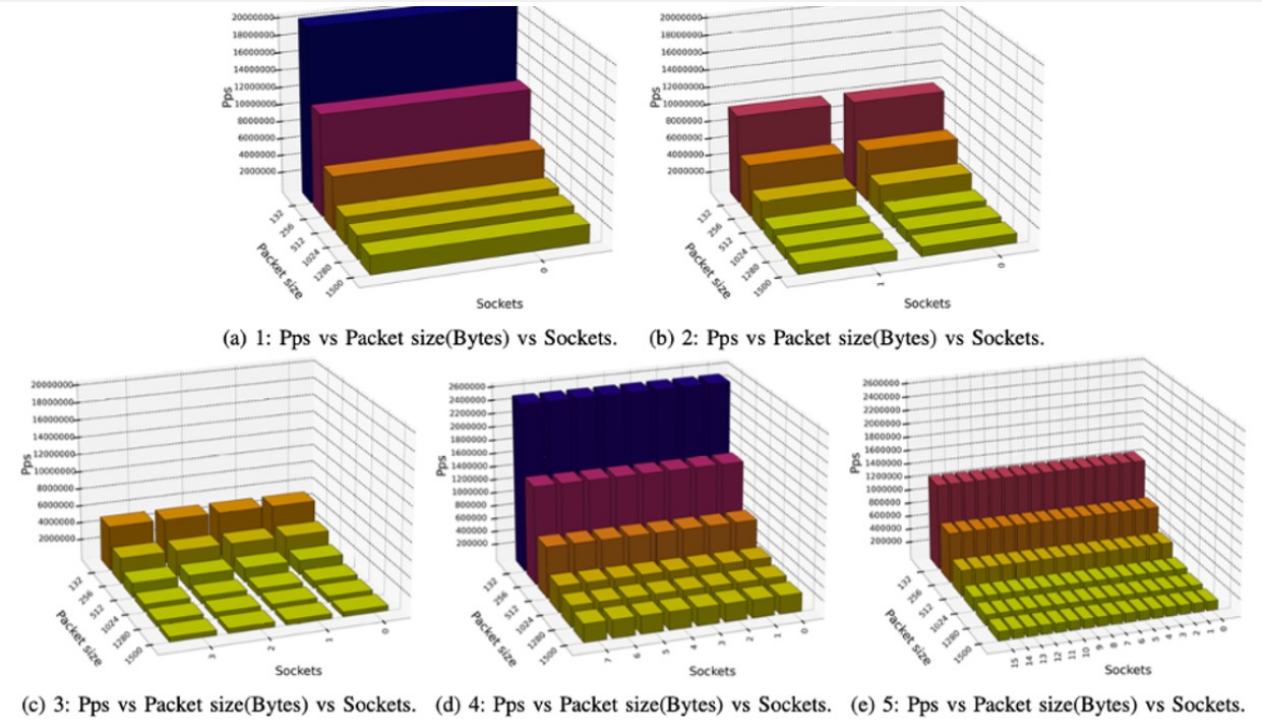
<소켓이 1개 일때>
패킷 크기가 132 바이트 일 경우
- 패킷 손실률은 70%
패킷 크기가 1024,1280,1500 바이트일 경우
- 0%에 수렴

<소켓이 2개 일때>
패킷 크기가 132 바이트 일 경우
- 패킷 손실률은 40%
패킷 크기가 512 바이트 이상일 경우
- 0%에 수렴

<소켓이 4개 일때>
최악의 조건
패킷 크기 - 132바이트
제어 규칙수 – 131072
- 0%

즉, 소켓 수 ⤴️ 패킷 손실률 ⬇️

대역폭



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

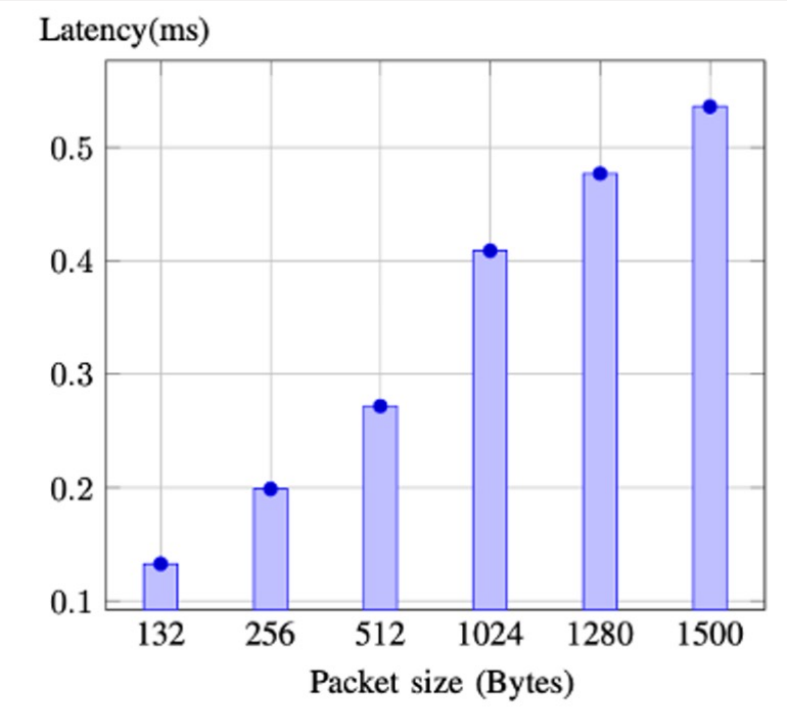
(c) 3: Pps vs Packet size(Bytes) vs Sockets. (d) 4: Pps vs Packet size(Bytes) vs Sockets. (e) 5: Pps vs Packet size(Bytes) vs Sockets.

그래프 (a) : 최악의 시나리오
패킷 크기가 132 바이트
각 소켓당 초당 처리되는 패킷 수 2000만개
➡️ 70% 손실

그래프 (c) : 최적의 시나리오
각 소켓당 500만개 씩 처리
➡️ 0% 손실

초당 처리되는 소켓의 패킷 수⤴️
패킷 손실⤴️

RTT


Latency(ms)
0.5
0.4
0.3
0.2
0.1
132
256
512
1024
1280
1500
Packet size (Bytes)

패킷 크기 ⤴️ RTT ⤴️

결론

eBPF/XDP 기반 네트워크 필터링은 멀티 테넌트 흐름을 동적으로 식별 및 분류 할 수 있으며, 빠른 트래픽 처리가 가능 + 패킷 필터링

eBPF 하드웨어 맵은 실시간 네트워크 메트릭을 기록하고 보고 및 메모리 접근 시간을 최적화하여 데이터 처리 속도 향상

Agilio CX SmartNIC을 사용하여 Linux 커널을 우회하여 하드웨어와 통신하게 하여 네트워크 성능 개선

즉, 미래의 6G의 까다로운 기준을 충족할 수 있는 확장성, 적합성을 보여준다.