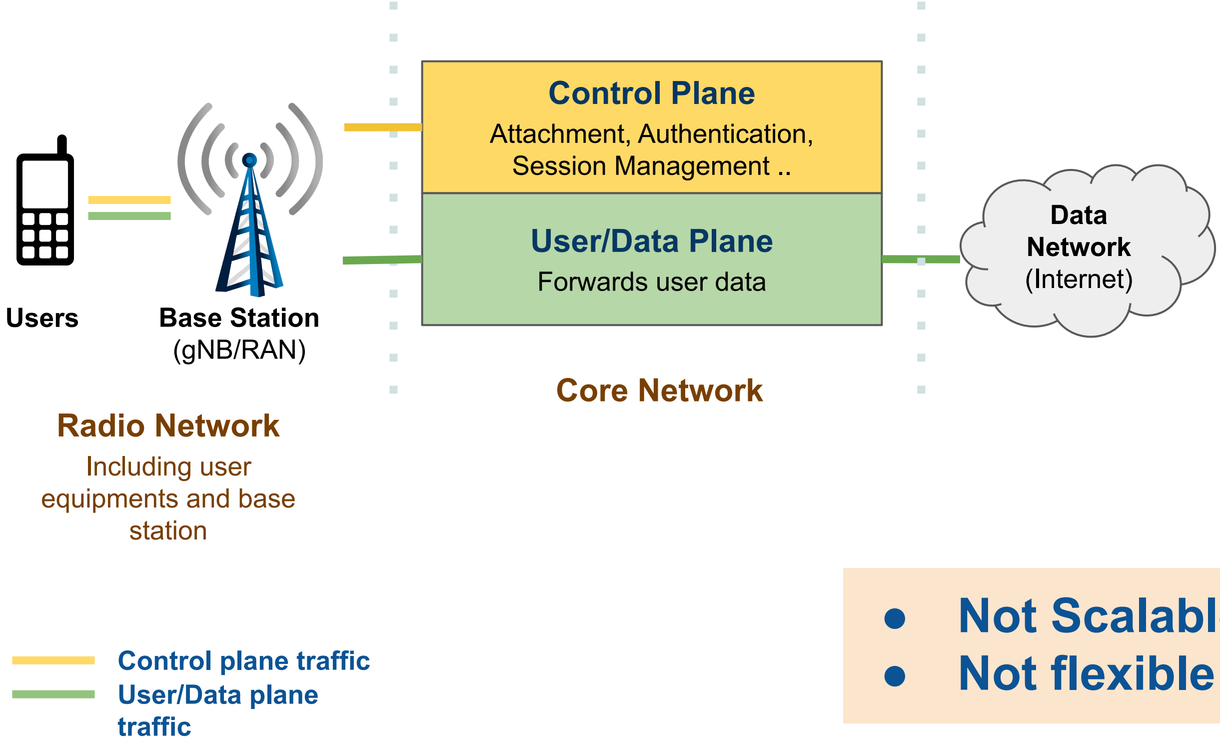
Leveraging Programmable Dataplanes for a High Performance 5G User Plane Function

Abhik Bose , Diptyaroop Maji , Prateek Agarwal, Nilesh Unhale, Rinku Shah, Mythili Vutukuru

Department of Computer Science & Engineering
Indian Institute of Technology Bombay

5th Asia-Pacific Workshop on Networking (APNet 2021)
June 24-25 2021, Shenzhen, China

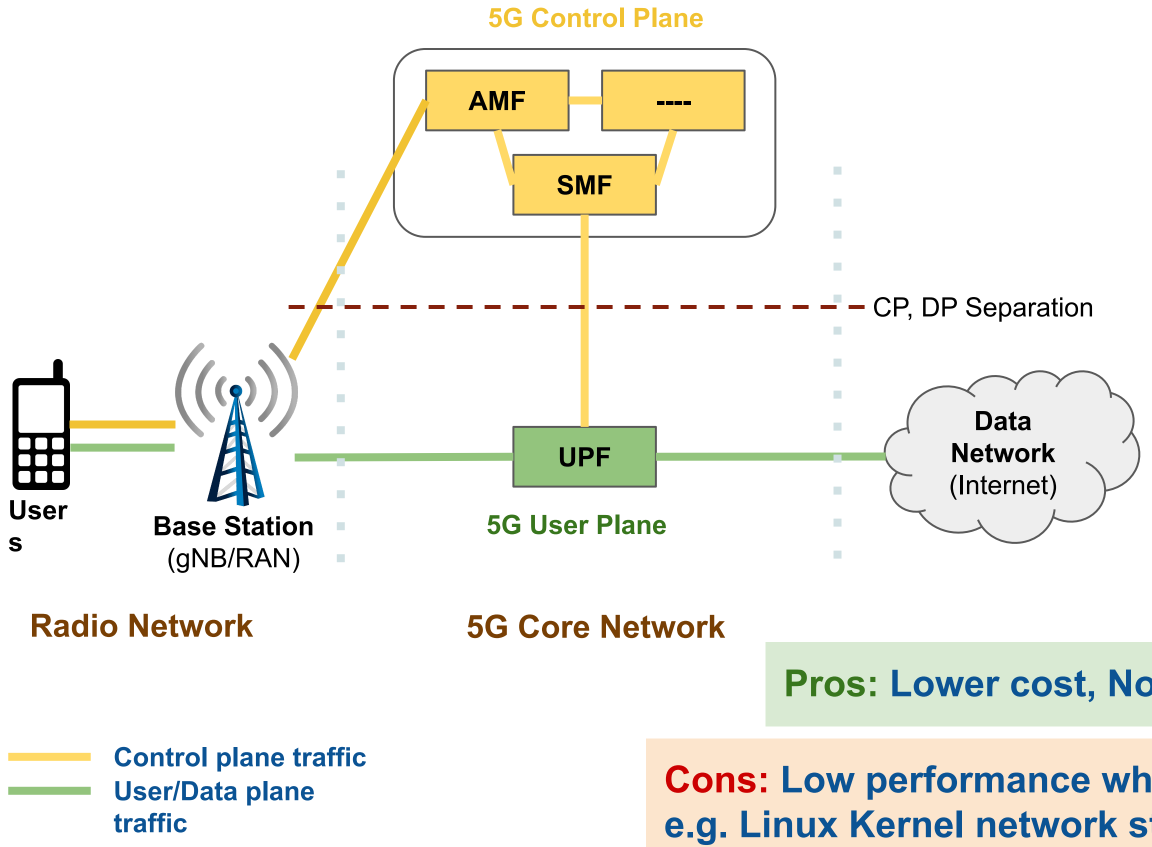
Traditional telecommunication network



- Specialised hardware
- Control and User plane in same box

- Not Scalable
- Not flexible

5G architecture, CUPS and NFV

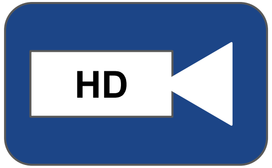


- **Control and User plane Separation (CUPS)**
- **Virtual Network Function (VNF) on commodity server**
- **CP NFs:**
  - **AMF:** Mobility
  - **SMF:** Session
  - **Other NFs:** Authentication, policy etc
- **User Plane Function (UPF):**
  - Forwards user data
  - Forwarding rules configured by SMF

Pros: Lower cost, No specialised hardware, scalable.

Cons: Low performance when using traditional network stack e.g. Linux Kernel network stack.

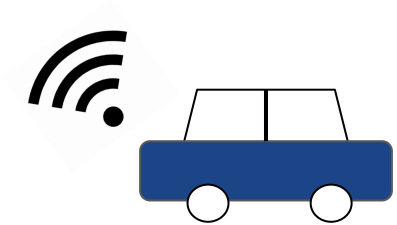
5G data plane requirements and use cases



High forwarding throughput

e.g. HD video streaming

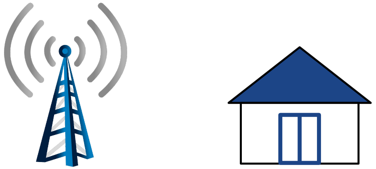
(~10 Gbps/km^2)



Low processing latency

e.g. Autonomous vehicles

(~1 ms)



Low-cost internet access

e.g. Internet in rural areas

How to meet UPF’s stringent 5G requirements?

Can state of the art UPF meet stringent 5G requirements?

Custom hardware NF

High performance software architecture

Metaswitch[1]
Intel / SK telecom[2]

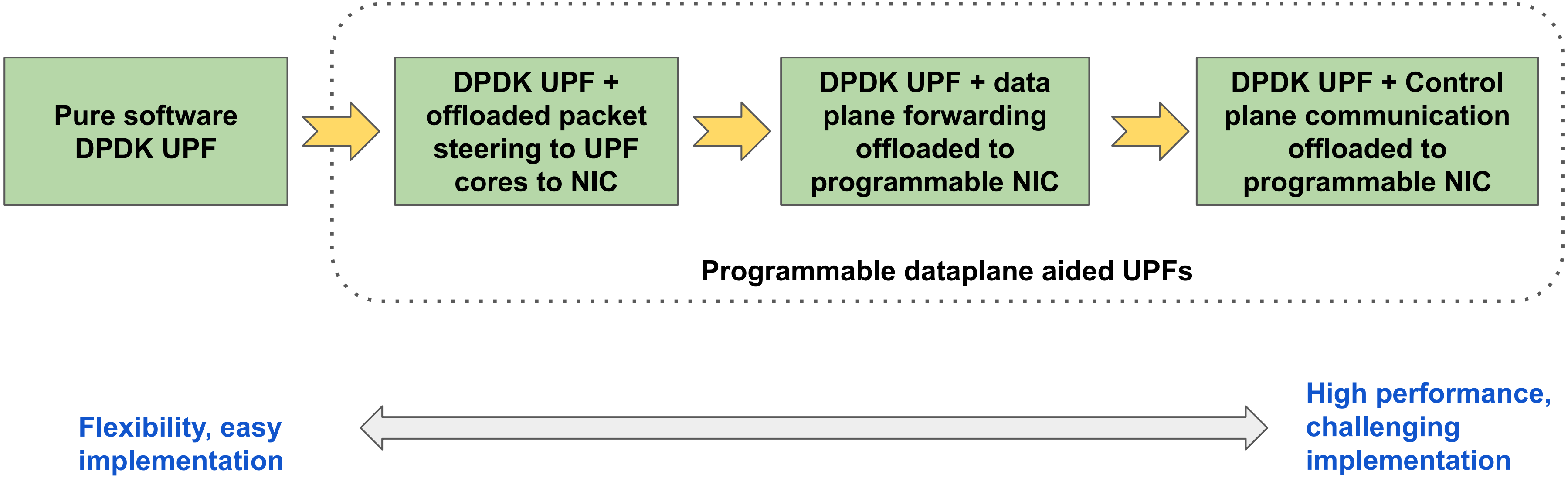
Offload to Programmable Hardware

Kaloom[3]
Mavenir[4]

- **What are all possible UPF functions that can be offloaded?**
- **What are the benefits of such offloads?**
- **No comparison across all possible offload solutions**

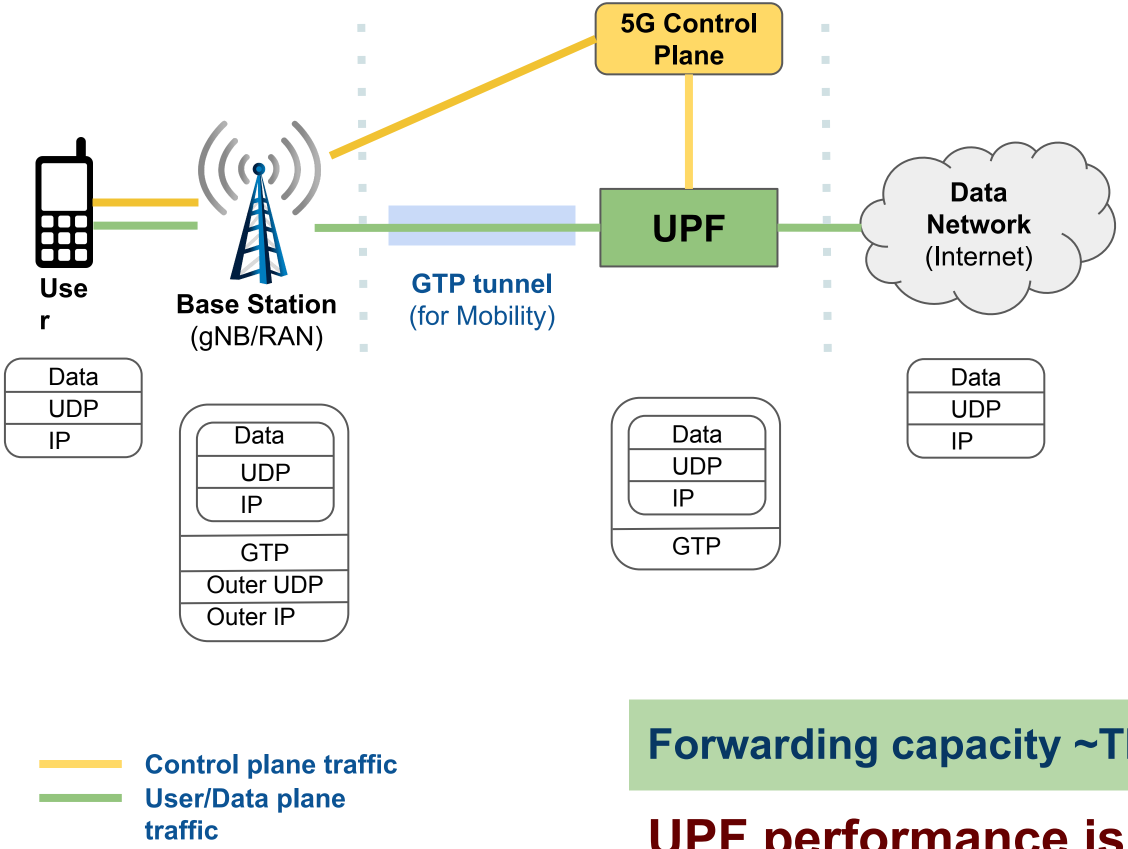
[1] Lighting Up the 5G Core with a High-Speed User Plane on Intel Architecture. (2019).
[2] DongJin Lee, JongHan Park, Chetan Hiremath, John Mangan, and Michael Lynch. Towards achieving high performance in 5G mobile packet core's user plane function. (2018).
[3] The Kaloom 5G User Plane Function (UPF). (2019)
[4] Optimizing UPF performance using SmartNIC offload. (2020).

Our contributions



- Evaluation of performance of all UPFs and comparison
  - Metrics: throughput, latency, cost/power efficiency
- Discussion of challenges in offloading UPF functionality
- Preliminary design of comprehensive offloaded UPF design

Background: 5G User Plane Function (UPF)

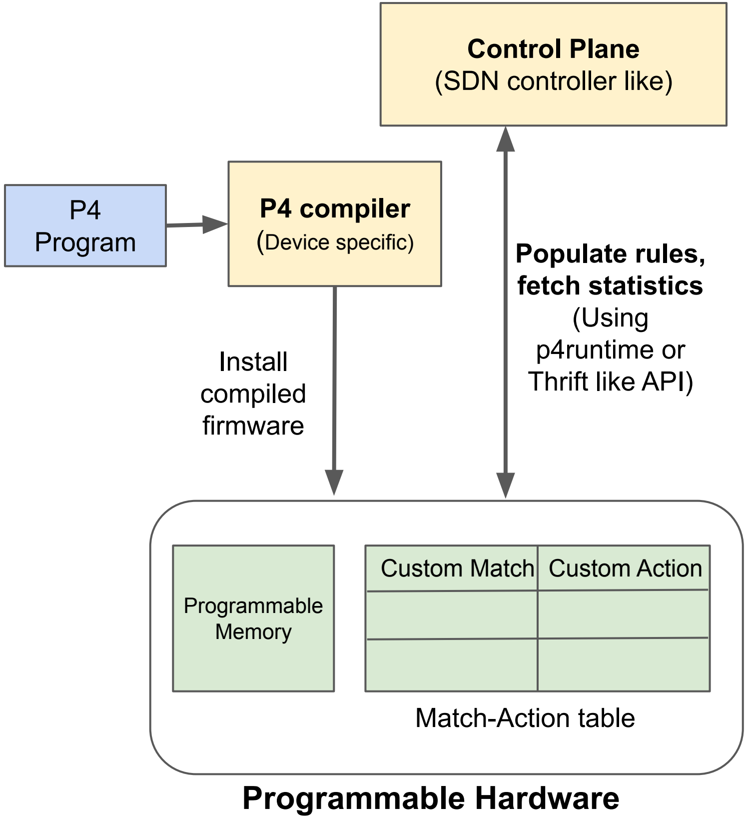


- **Control Plane communication**
  - Install session rules
- **Forwards user data**
  - Match packets against session rules
  - Forward, drop or buffer
- **GTP en/decapsulation**
- **QoS enforcement**
  - Rate limit per session
- Policy and Charging

Forwarding capacity ~Tbps. Critical for ultra low latency.

UPF performance is critical to future 5G success

Programmable data plane overview

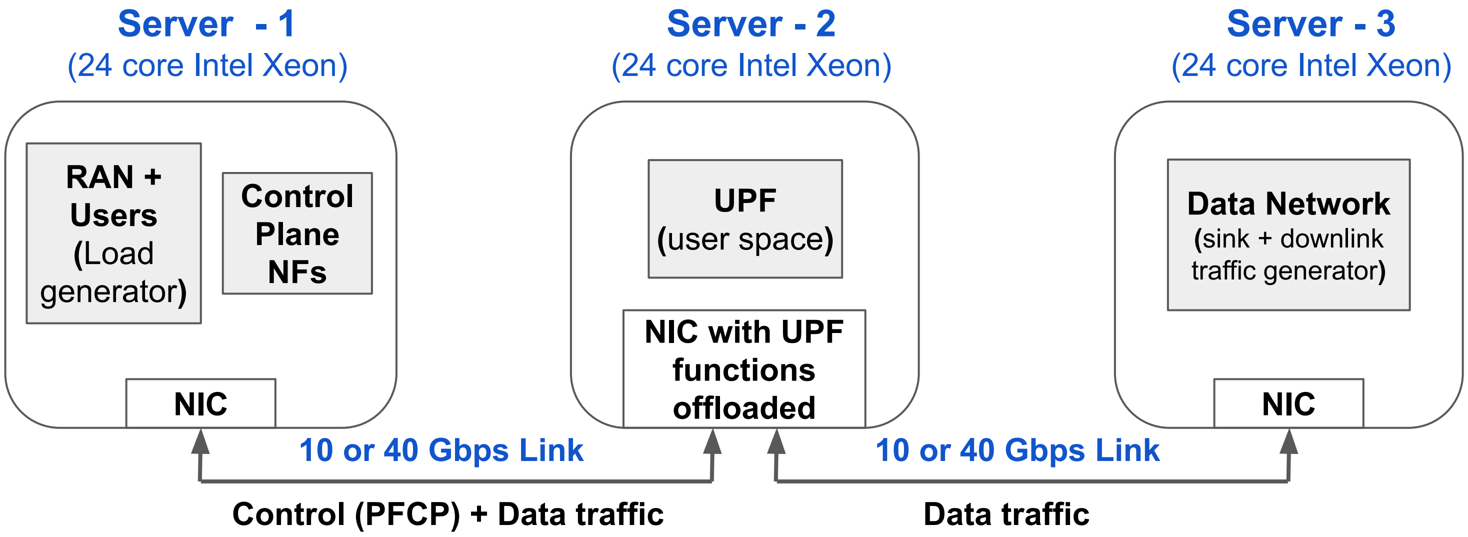


- **P4 Programmable hardware**
- **Features**
  - Custom Header parsing, custom match action
  - On-NIC programmable memory
  - Custom computation
  - Device specific features
- **P4 runtime or other APIs to configure custom match action tables at runtime**

Pros: Offloading application processing to programmable hardware is cost effective and improves performance

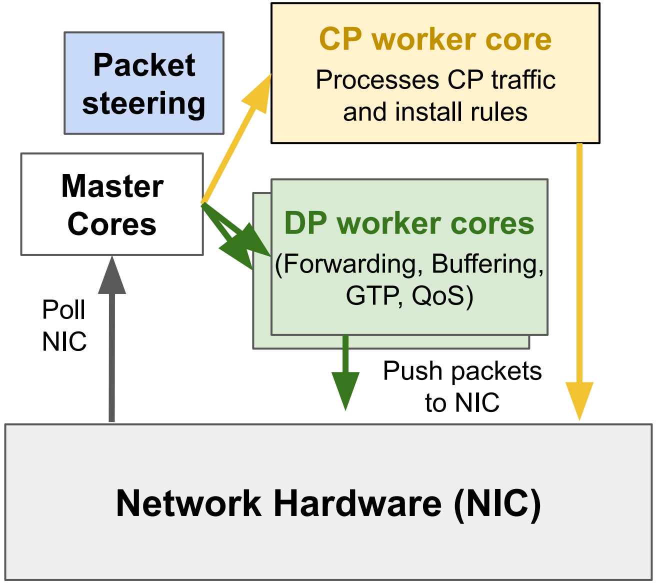
Limitations: Limited expressiveness, limited memory

Experimental setup for comparing UPF designs



- Agilio CX 2x10GbE programmable NIC for dataplane offloaded UPF
- XL710 i40e 40 Gbps NIC for packet steering offloaded UPF
- Load generator simulates control+data traffic from multiple users

Pure software DPDK UPF design



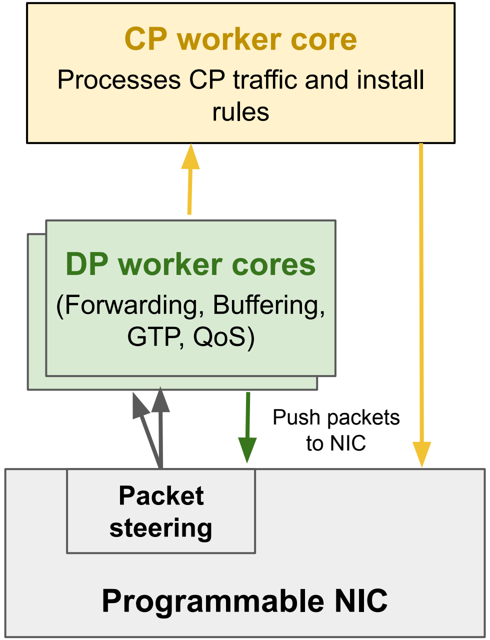
- **DPDK framework for high performance**
- **Multi-core scalable**
  - Master cores poll NIC, worker cores process packets
- **Purely software based**
  - Packet steering to worker cores, CP and DP processing
- **Packet steering**
  - Packets from same UE steered to same core, lockless

Pros: Scalable

Cons: High CPU usage. Higher cost.

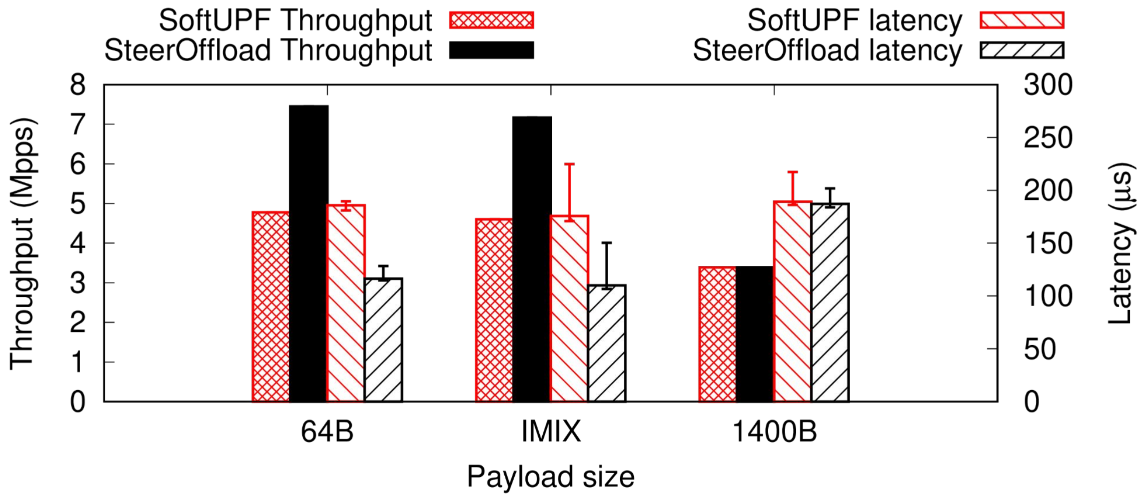
Software UPF Serves as performance baseline

Packet Steering Offload (SteerOffload) UPF design



- Regular NIC steer packets based on regular TCP/IP headers
  - Packets of a user can go to different cores or must be steered in software
- With programmable NIC, can parse user identifiers and redirect traffic of a user to specific core in hardware itself

SteerOffload: Forwarding throughput and latency improvement



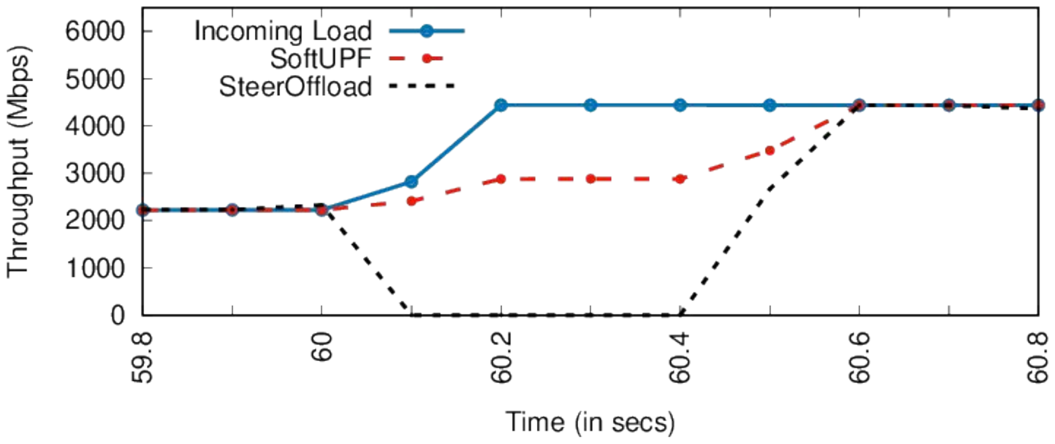
Pros: Offloading packet steering yields up to 45% higher throughput and up to 37% lower latency

- Avoiding packet steering offload in software

Is offloading packet steering always good?

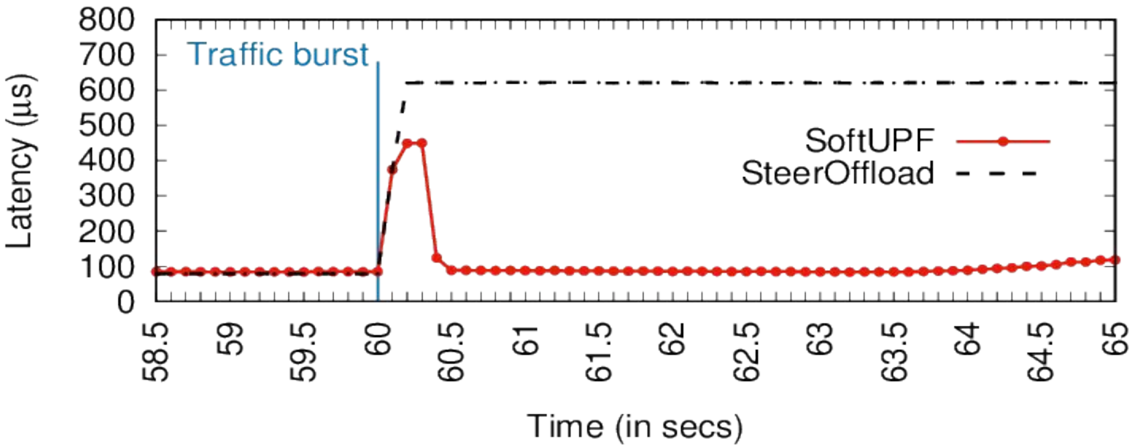
SteerOffload: Effect on dynamic scaling

Cons: Less flexible. NIC needs to be restarted for UE reassignment



- **Experiment: increase incoming load suddenly, dynamically scale UPF**
- **SoftUPF scaled with no downtime**
  - **Easily spawn worker threads**
- **SteerOffload UPF needs a NIC restart for scaling**
  - **Need to configure hardware queues**
- **SteerOffload took ~500 ms to scale**

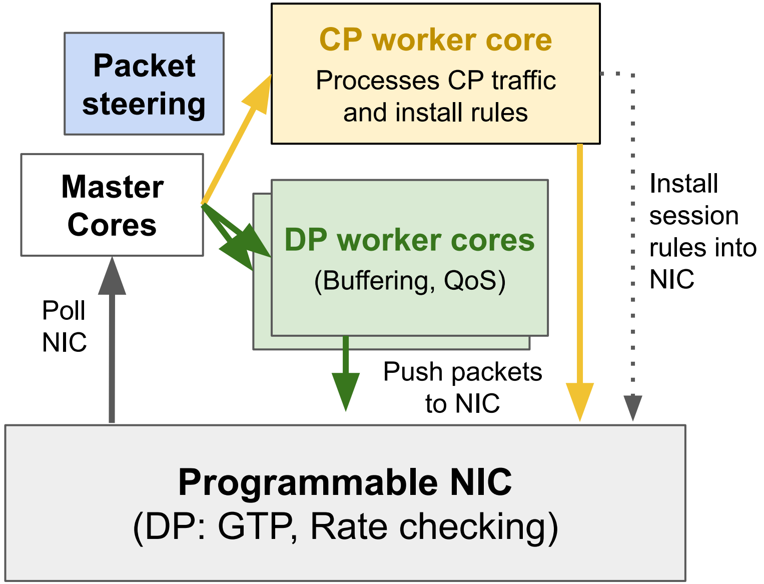
SteerOffload: Effect on heavy hitter UE



- **Experiment: single heavy hitter UE**
- **SoftUPF quickly re-distributes load among worker cores**
- **SteerOffload lacks ability to reconfigure packet steering based on load**
- **SteerOffload had 7 times more latency for the heavy hitter UE**

Conclusion: SteerOffload NOT suitable under dynamic and skewed workload

Data Plane offload (DPOffload) UPF design and benefits



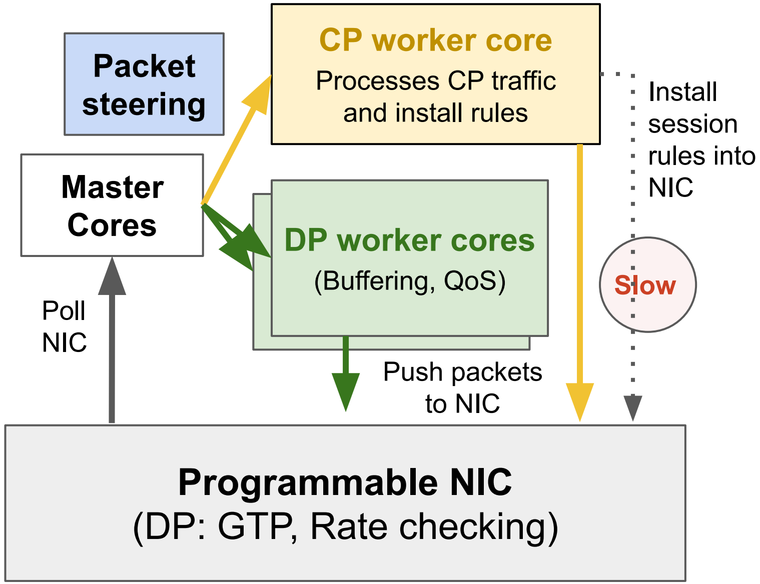
- **Offloaded:**
  - Session rule matching and forwarding
  - GTP en/decapsulation
  - Incoming rate verification using P4 meter
- **Oversubscribed flows are processed at user space**

UPF Design	64B packet	IMIX Packet	1400B packet
SoftUPF	138 uS	176 uS	294 uS
DPOffload	130 uS	140 uS	222 uS

Pros: DPOffload UPF has up to 24% lower latency

Control Plane performance penalty in DPOffload design

Is offloading packet steering always good?

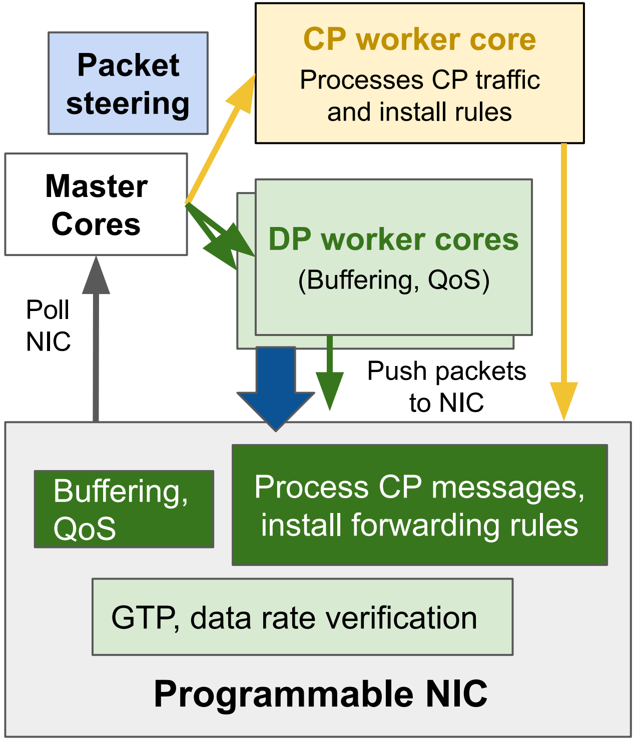


- **Data forwarding rules in hardware configured by controller software in userspace**
- **Slow control plane mediated session rule installation**
- **Bottleneck: Hardware - user space communication**

Performance metric	**SoftUPF**	**DPOffload**
Throughput (messages/sec)	5.1K	666
Latency (μS)	113	1646

DPOffload Cons: 86% lower control plane throughput and 15X higher control plane latency

Control Plane offload (CPOffload) design prototype



Solution:

- Process signaling messages from control plane also in hardware
- Install data forwarding rules from hardware itself

Challenges:

- Complex signaling packet format (variable length, recursive structure)

Our assumptions:

- Fixed packet format

Prototype design:

- Session rules in dataplane registers

Control Plane offload (CPOffload) design prototype

Performance metric	SoftUPF	DPOffload	CP Offload
Throughput (messages/sec)	5.1K	666	**2.05 M**
Latency (µS)	113	1646	**26**

Pros:

1. **402X and 3000X higher throughput compared to SoftUPF and DPOffload respectively**
2. **77% and 98% control plane latency reduction compared to SoftUPF and DPOffload respectively**

Summary

- UPF optimization is critical to 5G success.
- Offloading UPF functions to programmable hardware improves performance but decreases flexibility.
- Offloading data plane forwarding alone hurts capacity to process signaling messages that configure forwarding rules.
- Future work: Comprehensive UPF design that offloads both data plane forwarding and control plane communication processing to hardware.

Thank You!