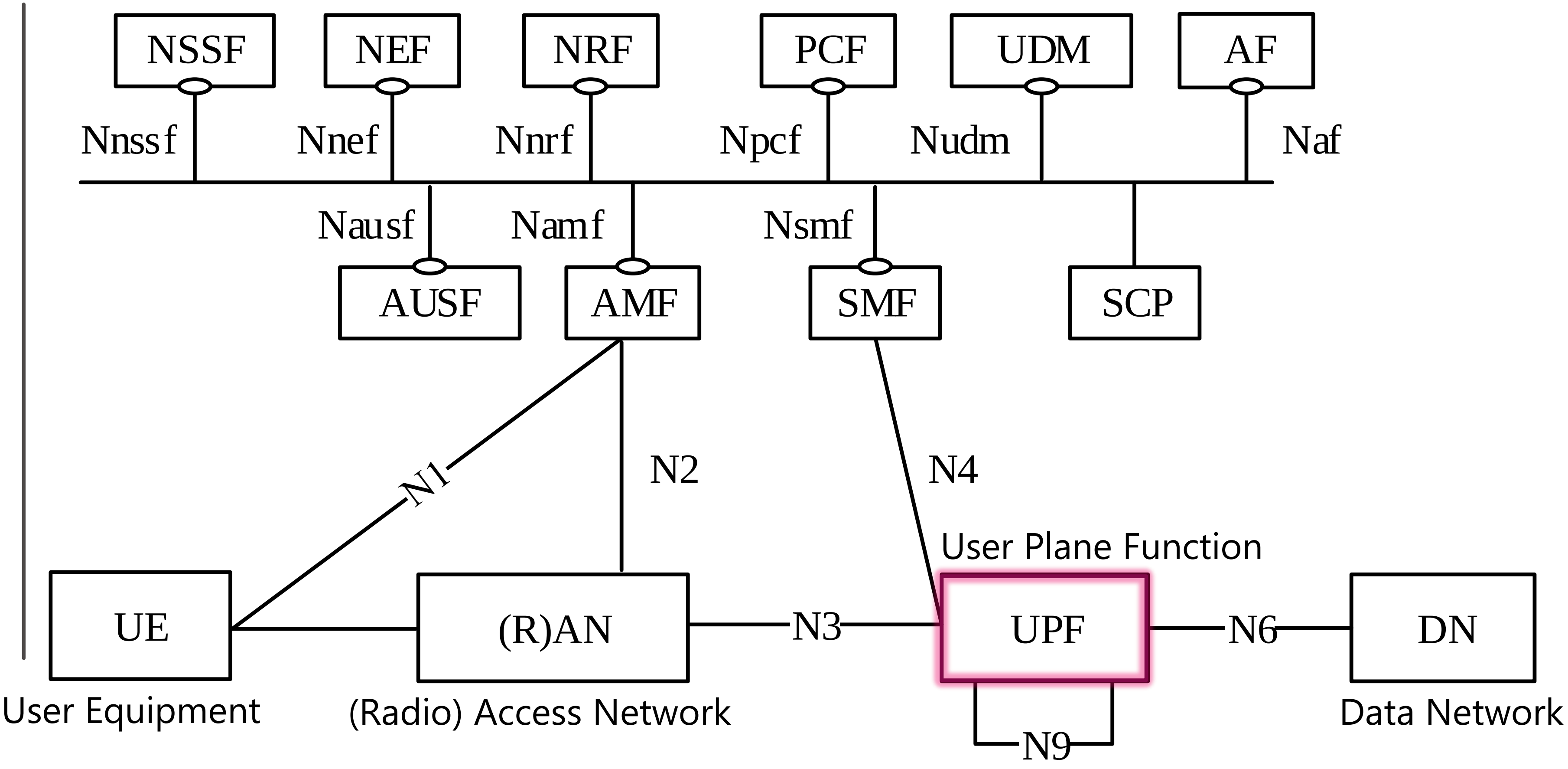
Using DPDK APIs as the I/F Between UPF-C and UPF-U

BARAK PERLMAN & BRIAN KLAFF
ETHERNITY NETWORKS

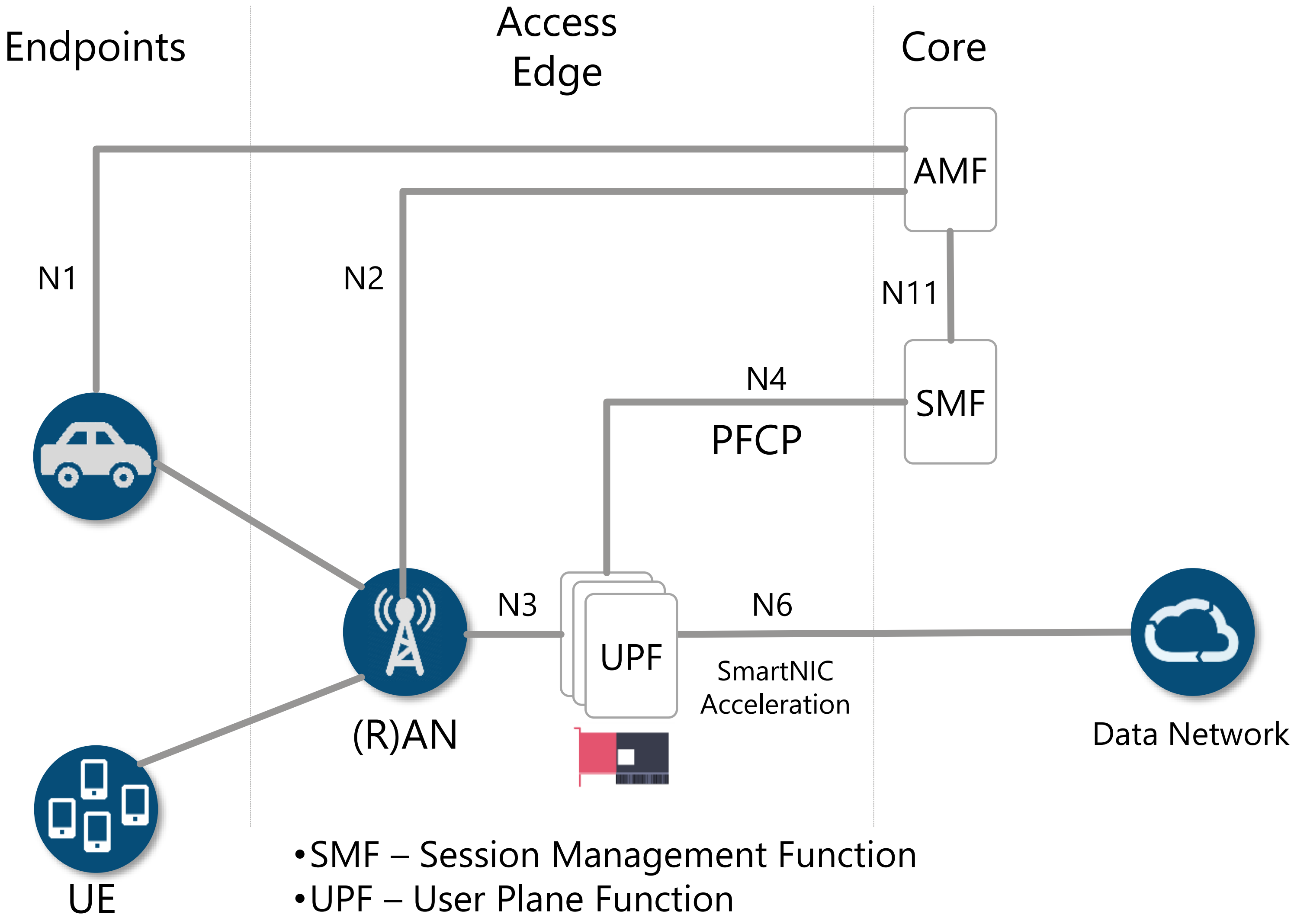
5G System Architecture

- UPF is a 5G architecture data plane element
- Replaces the user plane of SGW and PGW
- Control and User Plane Separation (CUPS)



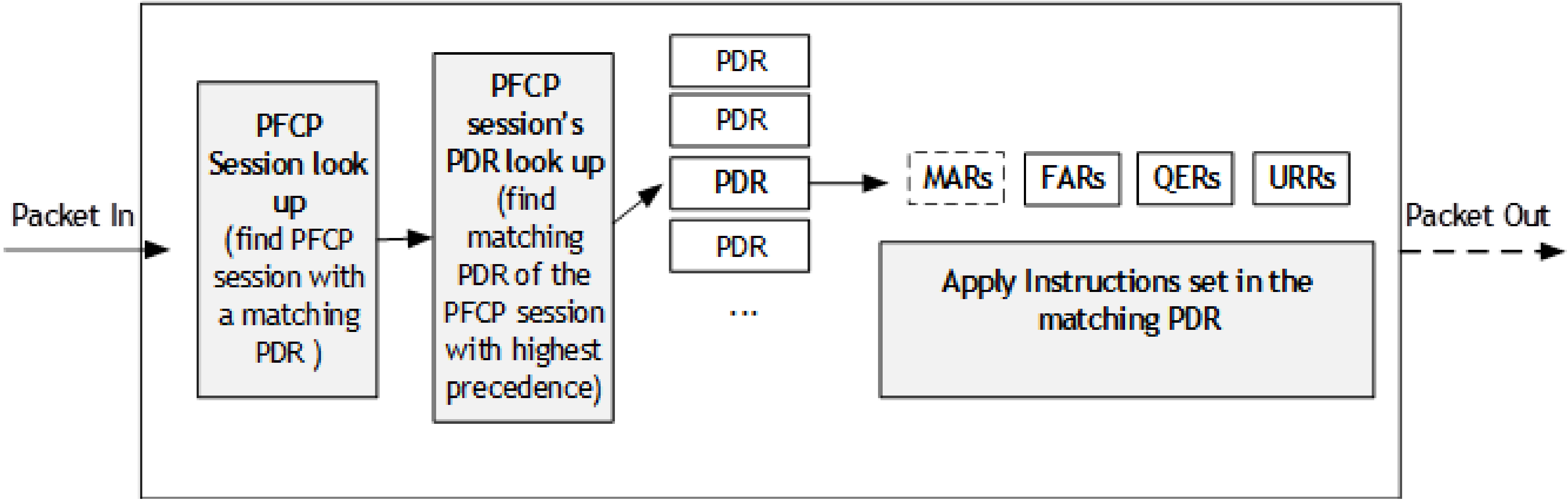
Accelerating UPF in 5G

- Many operators are now moving UPF to the edge
- Optimal UPF at aggregation locations
- Used for local breakout
- Partial/complete data plane offloading over FPGA-based SmartNICs
  - Programmable
  - Scalable
  - Open APIs



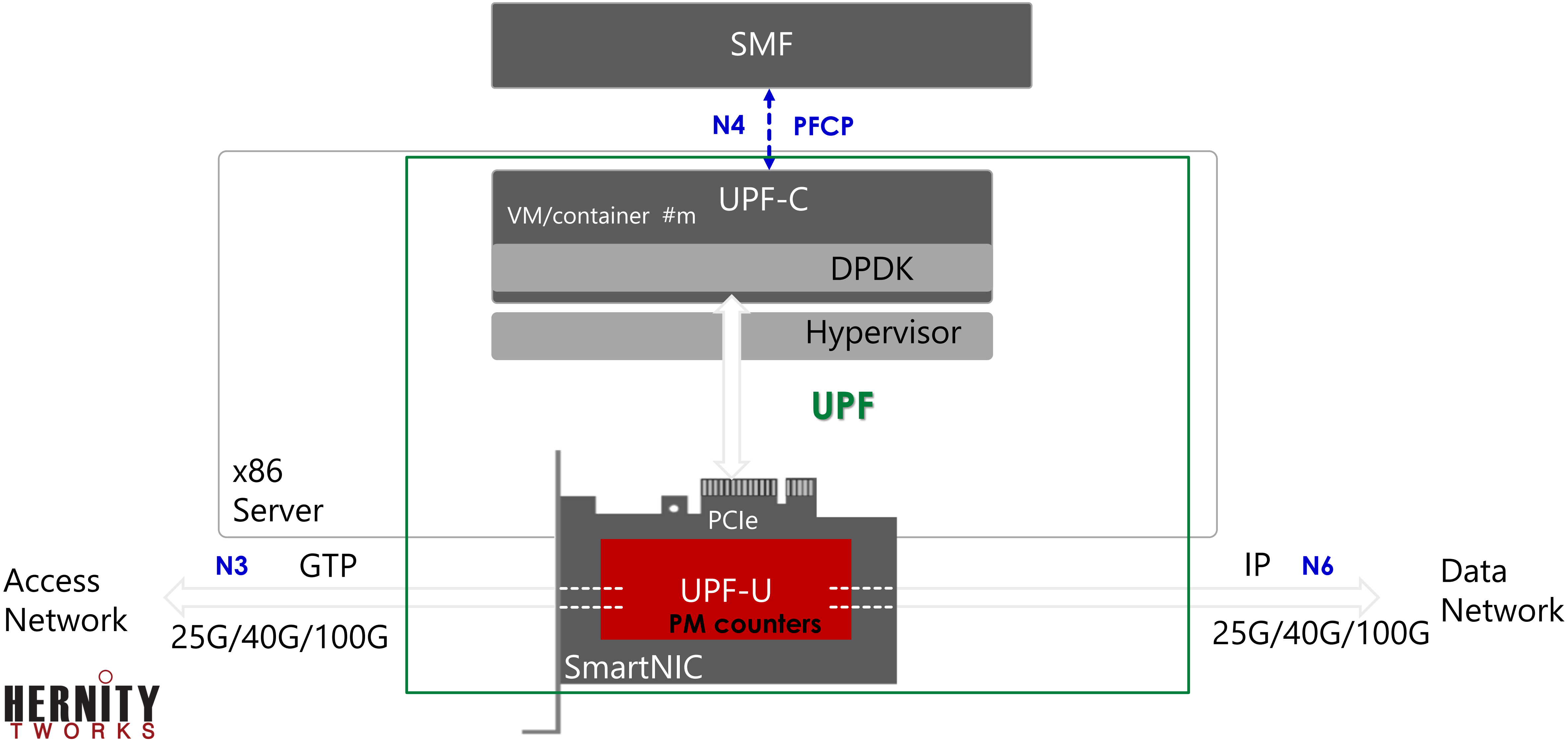
- SMF – Session Management Function
- UPF – User Plane Function
- PFCP – Packet Forwarding Control Protocol

Packet Processing Flow in UPF

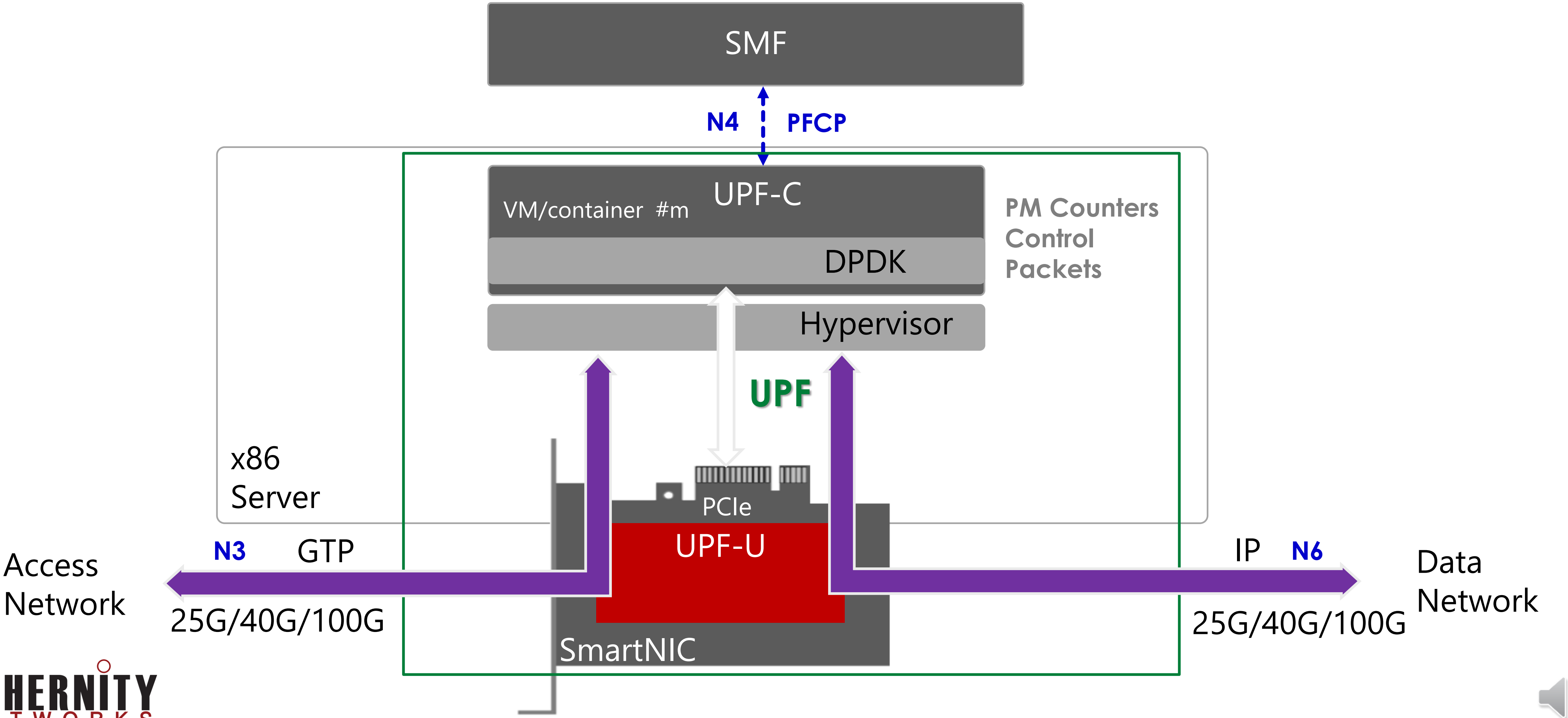

Packet In
PFCP Session look up (find PFCP session with a matching PDR)
PFCP session's PDR look up (find matching PDR of the PFCP session with highest precedence)
PDR
PDR
PDR
PDR
...
MARs
FARs
QERs
URRs
Apply Instructions set in the matching PDR
Packet Out

PDR – Packet Detection Rule
FAR – Forwarding Action Rule
QER – QoS Enforcement Rule
URR – Usage Reporting Rule
BAR – Buffering Action Rule
MAR – Multi-Access Rule

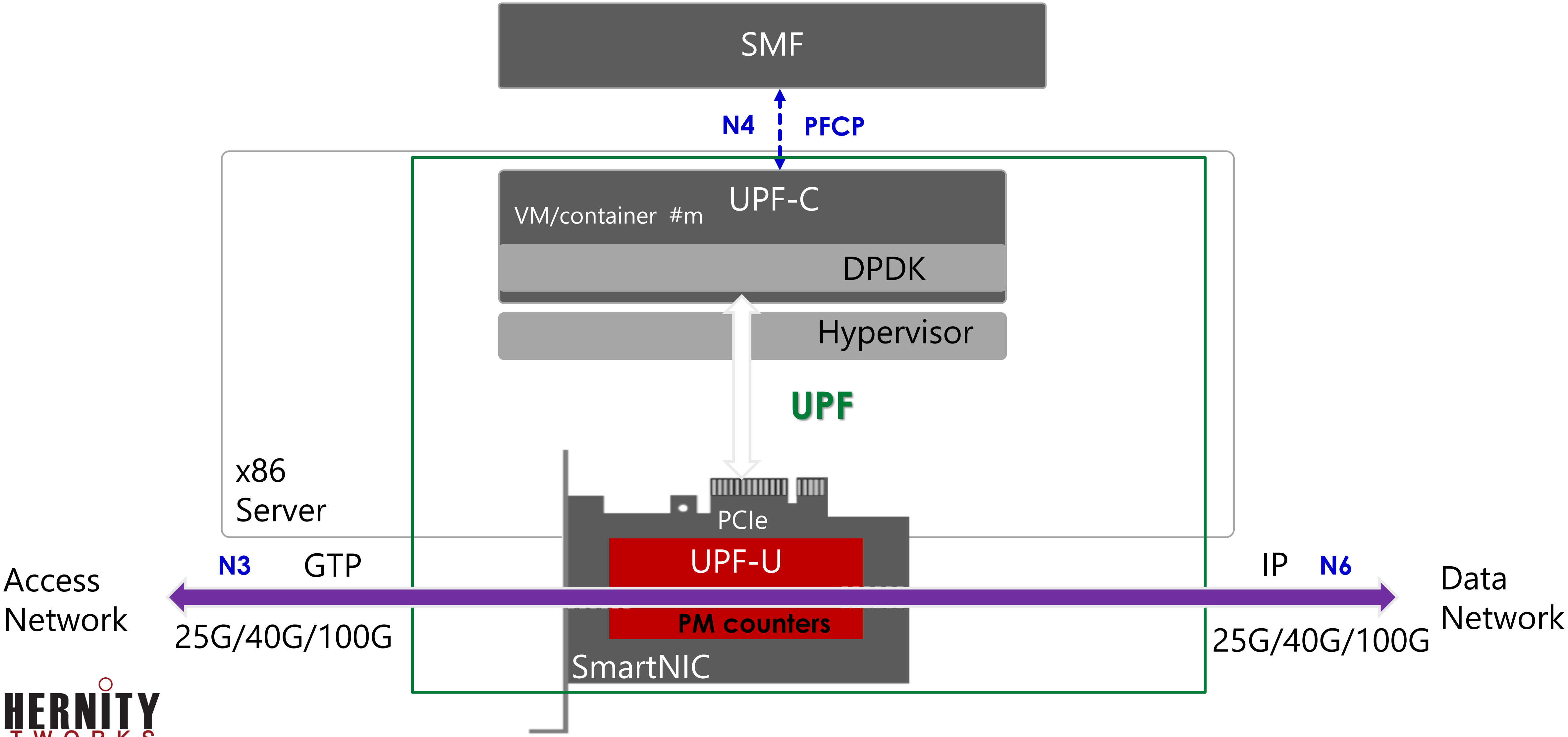
Separation of UPF to UPF-C and UPF-U



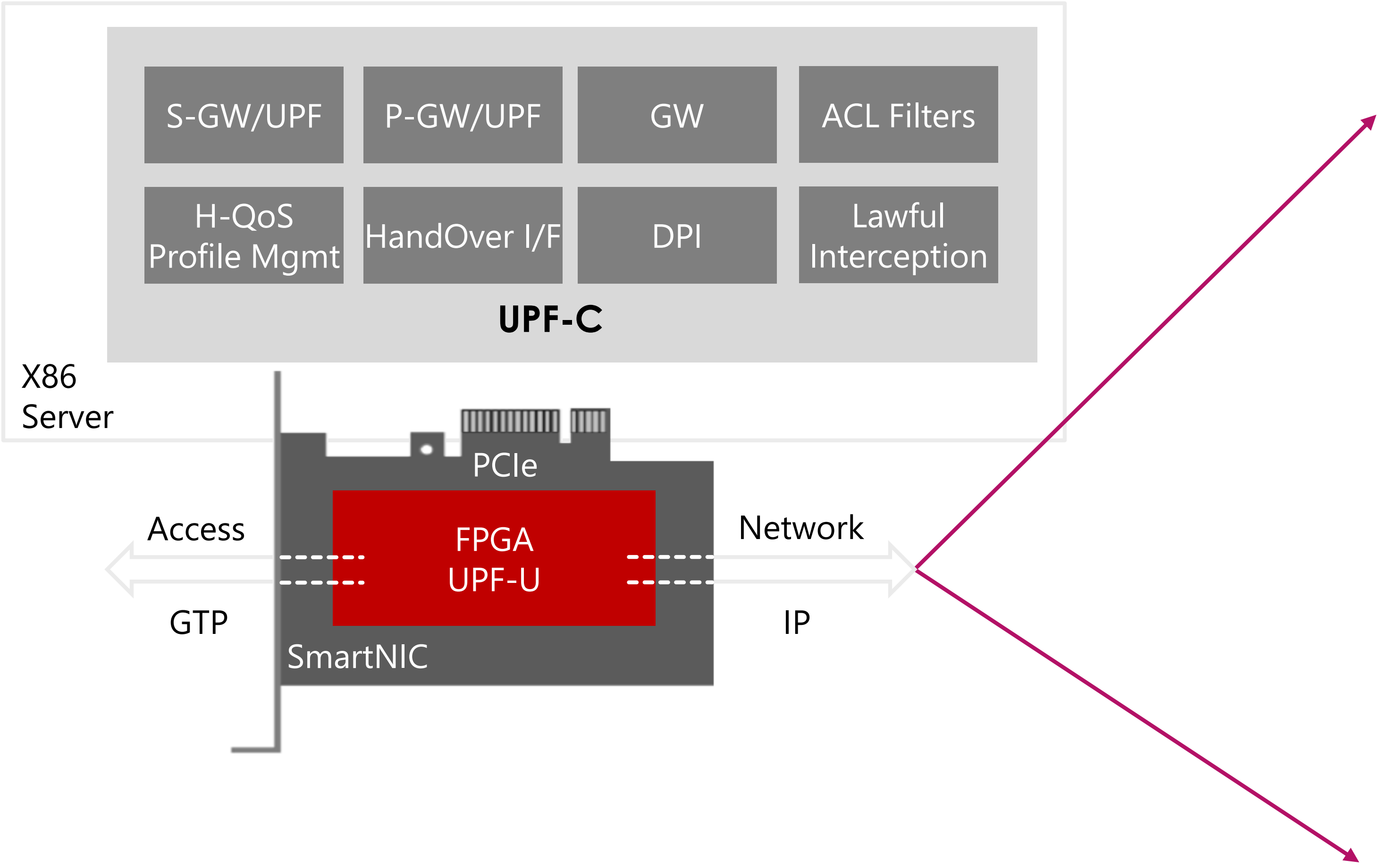
Partial Offload



Full Offload



FPGA SmartNIC Accelerates UPF Features

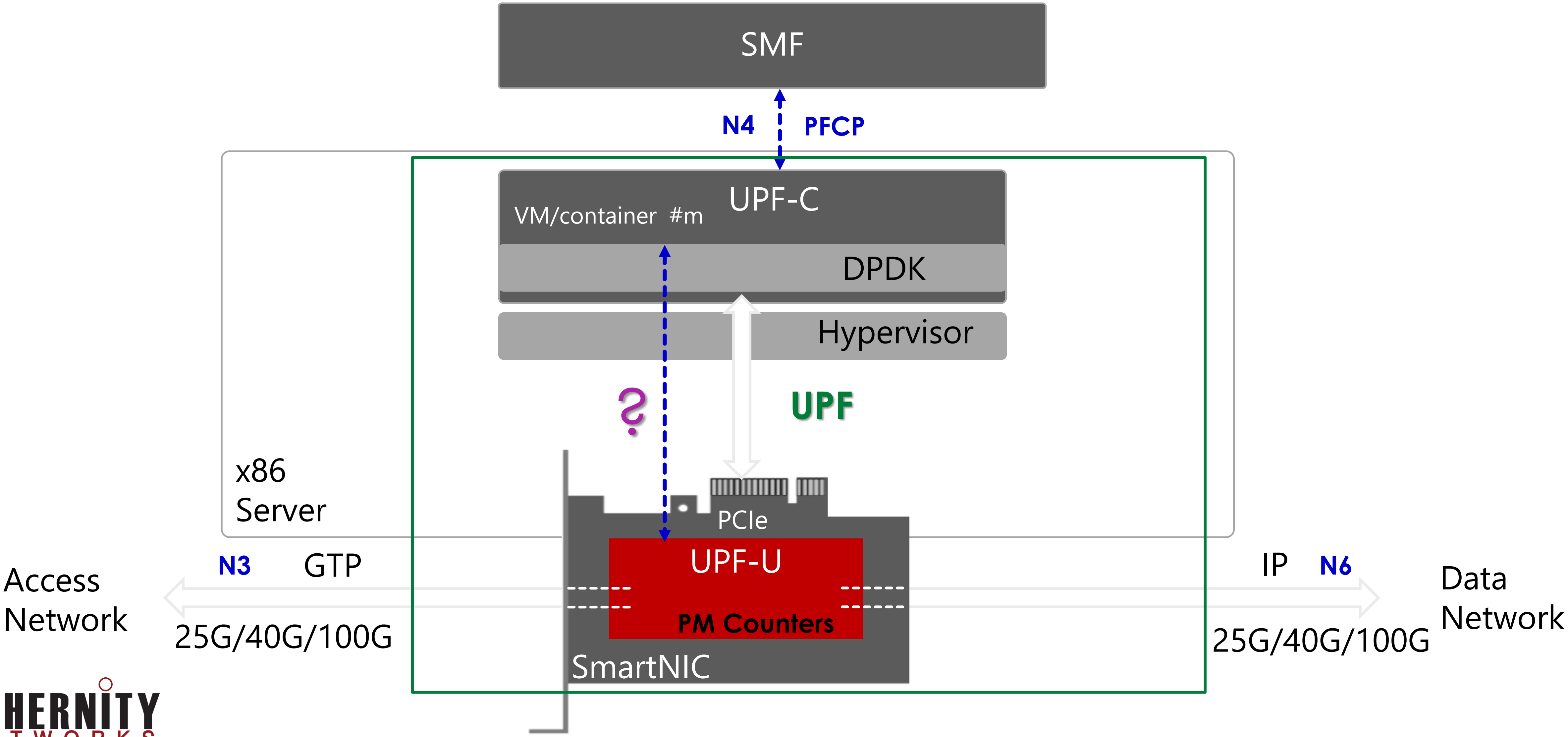


UPF-U features offloaded by SmartNIC

- Packet routing forwarding
- GTP termination (if needed)
- Gating, redirection & traffic steering
- QoS
- Packet buffering
- Packet duplication
- ACL
- Lawful interception
- PM counters collection for billing
- IPsec encryption & decryption (for N3IWF)

SmartNIC Forwarding Engine

I/F Between UPF-C and UPF-U

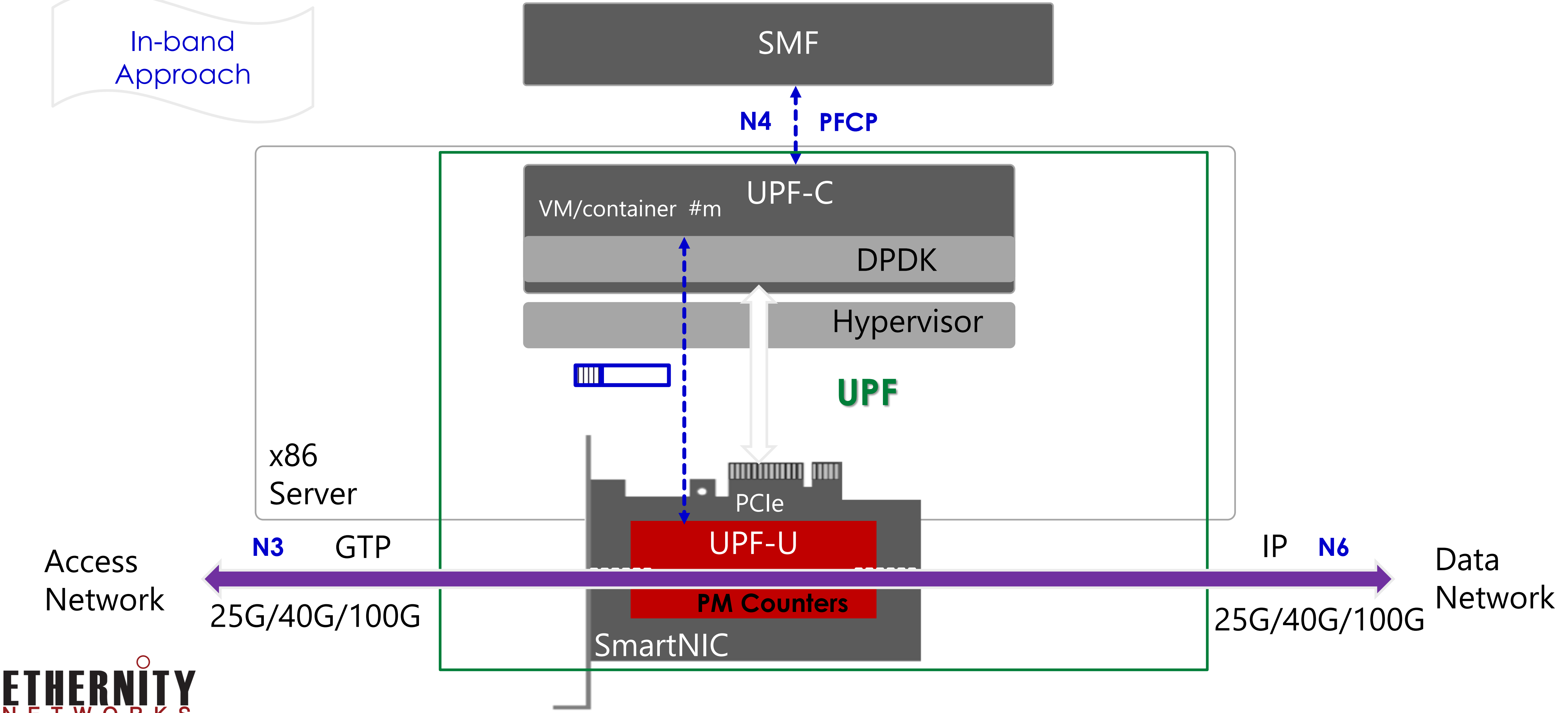


Options for UPF-C to UPF-U I/F

- Control Plane Messages
  - Send in-band control packets between UPF-C and UPF-U
  - Option 1: dedicated control packets for this purpose, new standard
  - Option 2: use an existing SDN I/F (for example, OpenFlow, P4 & P4 run-time)
- Use DPDK HW offload APIs
  - Use existing DPDK methods for HW offload

Control Plane Messages

In-band
Approach

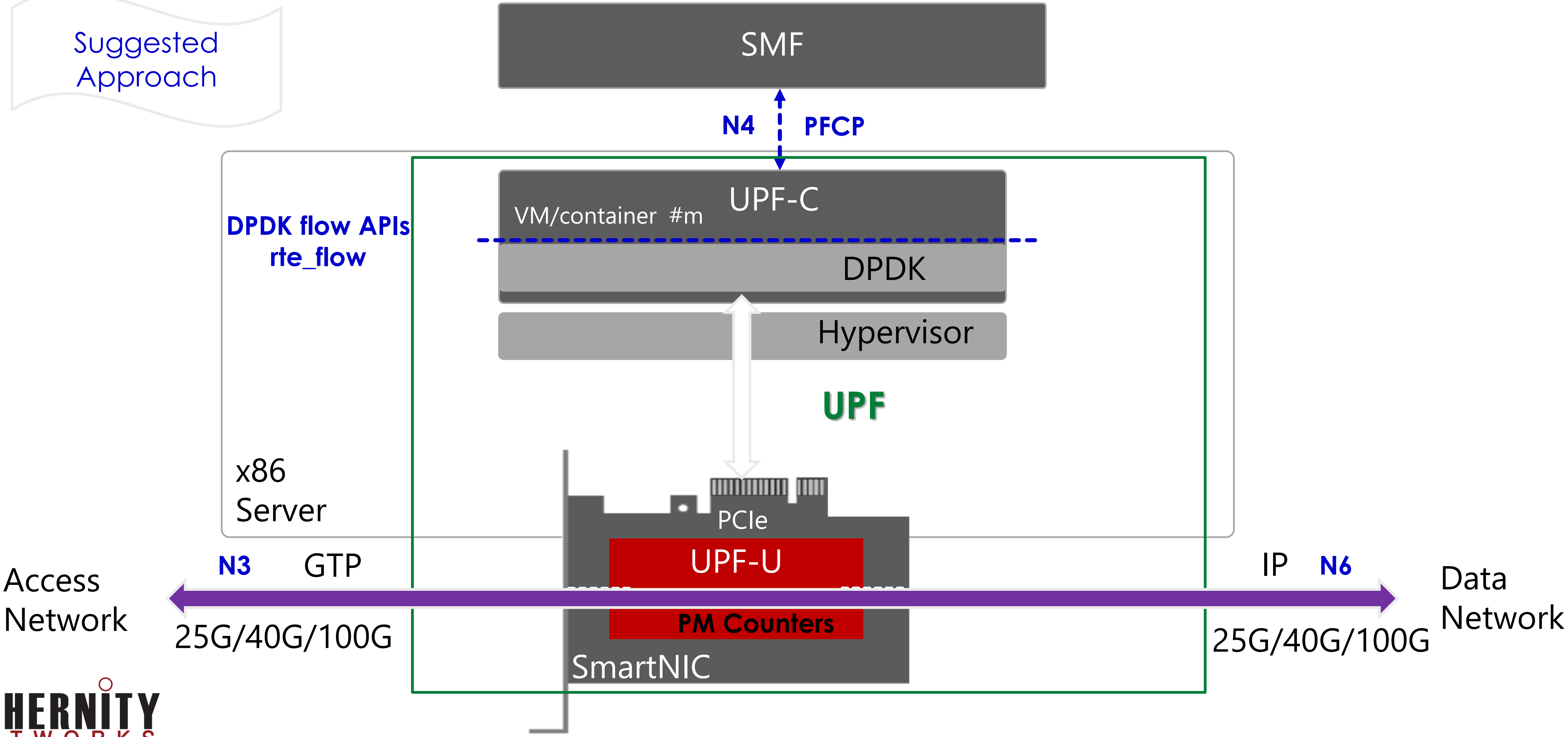


Dedicated Control Plane Messages

- Benefit: good performance
  - Control packets consume a small portion of the large data plane packets
- Dedicated Control Plane Messages
  - Need to define a spec for control plane message content for all UPF-U features
  - Need to implement a specific design in both UPF-C and UPF-U
  - Need to update the spec and implementation for new UPF-U features
  - Need to address error reporting and retransmission
- Existing SDN I/F
  - Need to adapt existing I/F to cover all UPF features not easily covered by existing SDN protocols
  - For example: cover policies, billing reports, etc.

Using DPDK HW Offload APIs

Suggested Approach



DPDK-Based APIs for UPF-C to UPF-U

- Most UPF applications are already implemented in DPDK
  - For example, 5G UPF based on VPP: https://github.com/travelping/vpp
  - rte_flow is the natural choice for DPDK applications
  - UPF is flow based, maps nicely to DPDK rte_flow offload APIs (generic flow API)
- Avoids vendor lock-in
  - Supported by a large variety of vendors
  - Becoming a de-facto standard
- Futureproof: maintained and enhanced by the DPDK community
- Provides methods for handling flow validation
- Flexible enough to cover almost all UPF-U features

DPDK rte_flow

SUGGESTED
IMPROVEMENTS REQUIRED
FOR UPF OFFLOAD

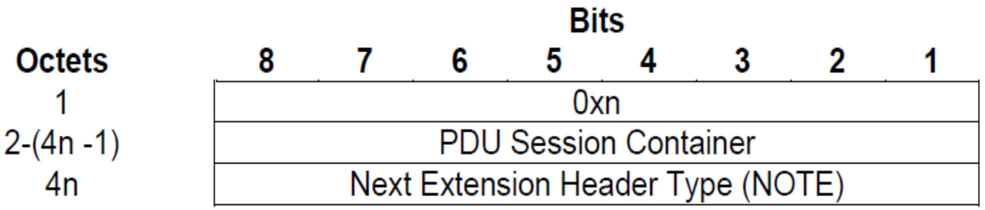
Improve DPDK rte_flow APIs Performance

- UPF requires a large number of flows (e.g., 1M flows)
  - Need to improve the rte_flow configuration rate
- Add burst write configurations
  - Batching of rte_flow entries and then committing the batch to the HW offload
  - Use shared memory and DMA for flow data structures
  - Provide pointers to complete rte_flow data structures
  - This is required for delivering PM counters for a large number of rte_flows
- Create a single rte_flow template, then populate just a few variable fields
  - Avoids configuration of repeated fields in the same rte_flow template

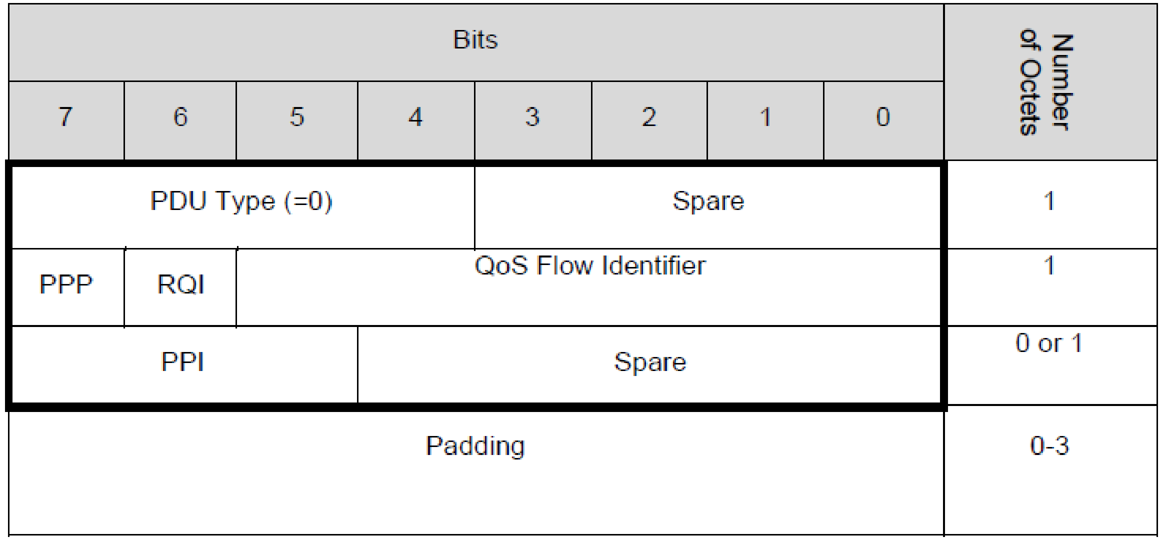
GTP Header

- GTP header match is already supported in rte_flow
- Need to add GTP-U encap/decap
  - GTP-U header encapsulation and decapsulation rte_flow actions
  - Very similar to other tunnel headers that are already supported: VxLAN, NVGRE, MPLS and raw_encap/decap
  - Should include optional support for 5GS GTP-U extension header
    - The GTP-U Extension Header for 5GS is called "PDU Session Container"

GTP-U Extension Header for 5GS

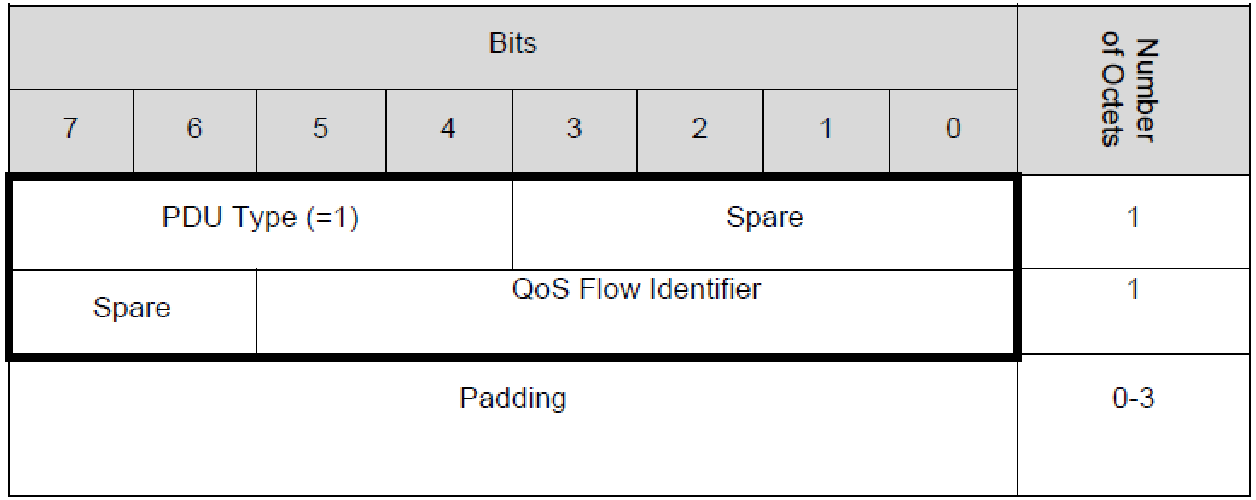


Octets	Bits							
	8	7	6	5	4	3	2	1
1	0xn							
2-(4n -1)	PDU Session Container							
4n	Next Extension Header Type (NOTE)							



Bits								Number of Octets
7	6	5	4	3	2	1	0	
PDU Type (=0)				Spare				1
PPP	RQI	QoS Flow Identifier						1
PPI			Spare					0 or 1
Padding								0-3

Figure 5.5.2.1-1: DL PDU SESSION INFORMATION (PDU Type 0) Format



Bits								Number of Octets
7	6	5	4	3	2	1	0	
PDU Type (=1)				Spare				1
Spare		QoS Flow Identifier						1
Padding								0-3

Figure 5.5.2.2-1: UL PDU SESSION INFORMATION (PDU Type 1) Format

Summary

- UPF is the 5G element implementing user plane data path
- UPF can be placed at the edge
- There is a need for UPF HW offload
- UPF can be split into UPF-C and UPF-U
- Need to define an I/F between UPF-C and UPF-U
- DPDK rte_flow APIs are a good option for implementing this I/F
- Need to implement some enhancements in rte_flow for optimal support of UPF

DPDK DATA PLANE DEVELOPMENT KIT

ETHERNITY NETWORKS

Thank You

BARAK PERLMAN, CTO
BARAK@ETHERNITYNET.COM

BRIAN KLAFF, MARKETING DIRECTOR
BRIANK@ETHERNITYNET.COM