How NICs Work Today

Tom Herbert, Intel
Simon Horman, Netronome
Andy Gospodarek, Broadcom

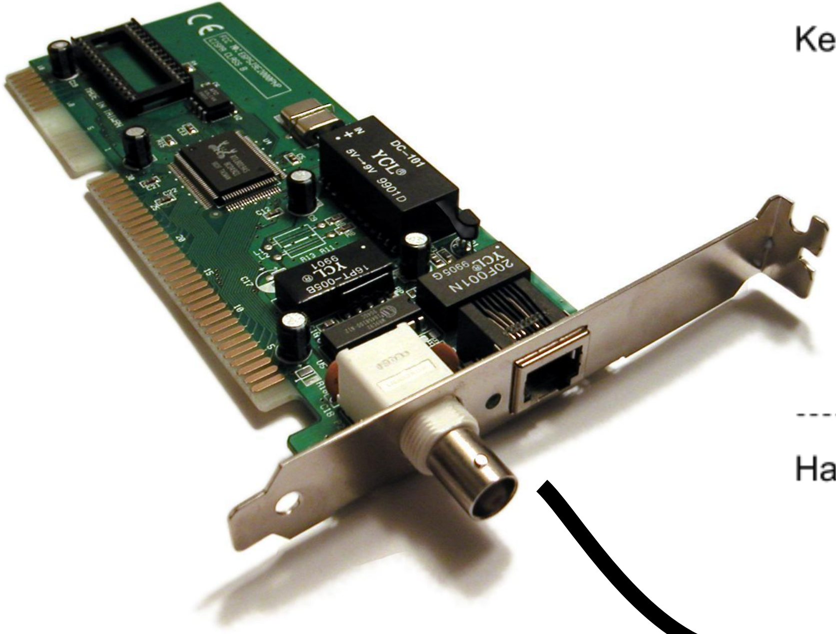
IETF 105, Montreal, Tuesday July 23, 2019

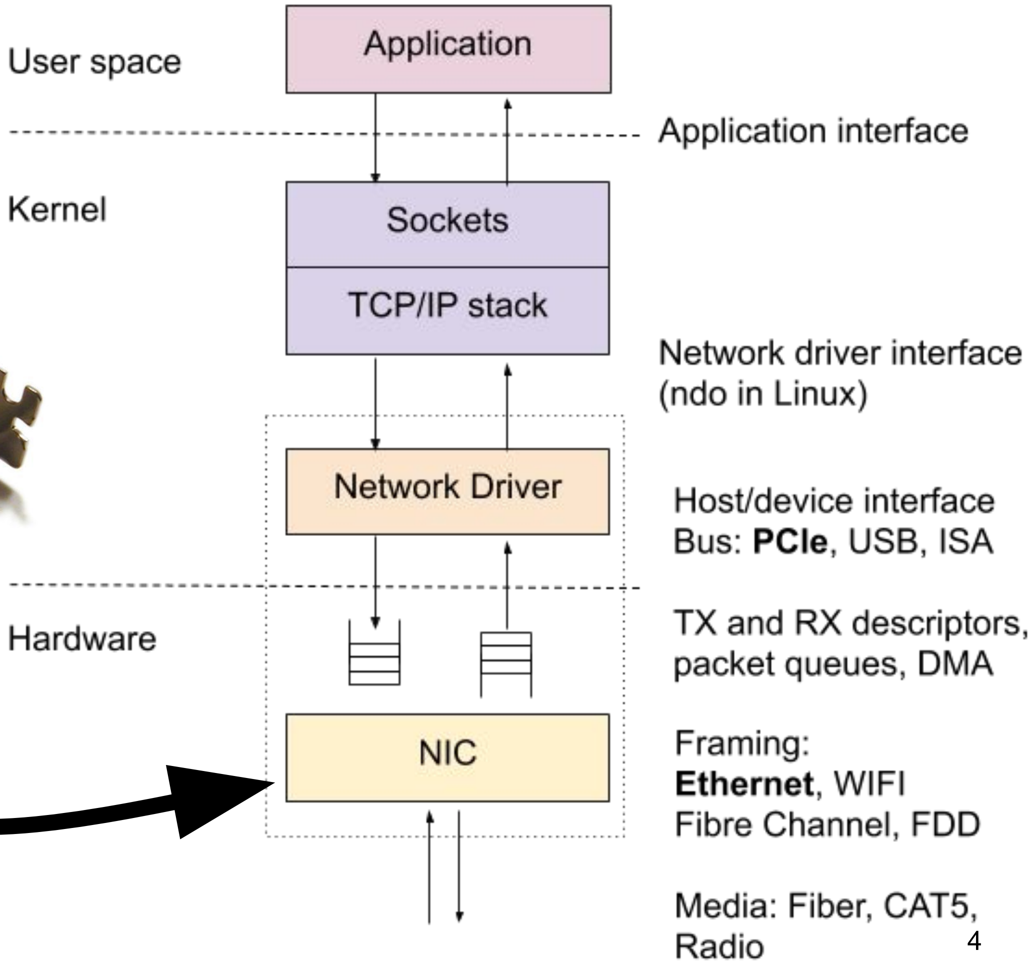
Fundamentals

Terminology

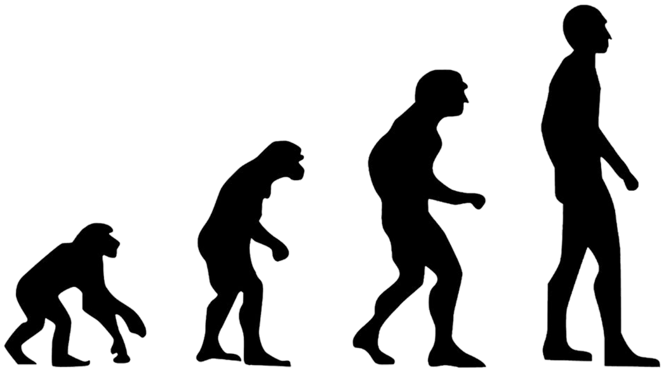
- **Network Interface Card (NIC):** Host's interface to physical network
- **Host Stack:** Software stack that performs host side processing of L2, L3, or L4 protocols
- **Kernel Stack:** Host stack implemented in an OS kernel
- **Offload:** Do something in NIC HW that could be done in host SW stack
- **Acceleration:** Offload for performance gains

Network Interface Cards





Evolution of Network Interface Cards



- Fundamental support (1990s)
  - Transmit and receive packets
  - Basic offloads (Ethernet Checksum Offload!)
- Data plane acceleration (early to mid 2000s)
  - Optimization for multi-core CPUs
  - Hardware data plane offload — mostly fixed function devices
  - Tunneling, IPsec, QoS offloads
- Programmability (2010 onwards)
  - FPGAs and NPUs with programmable data plane
  - General purpose processor with programmable data and control planes

Offload: Motivation

- Free up CPU cycles for application
- Specialized processing can be more efficient
- Save host resources
- Scaling performance (low latency/high throughput)
- Power savings for some use cases

=> Reduced TCO (*marketing slant!*)

Less is More Principle

- Protocol agnostic is better than protocol specific
  - Avoid protocol ossification
  - New protocol support without needing completely new solutions
- Common open APIs are better than proprietary ones
  - Avoid vendor lock in
  - Differentiation by features, performance, implementation
- Programmability is (generally) good
  - Be adaptable, don’t dictate to the user what they are allowed to do
  - Aspiration: “write once, run anywhere” model across devices

Basic offloads

Offload Considerations

- TX and RX
- Protocol agnostic versus protocol specific
- Stateful versus stateless
- Encapsulation
- “Always on” versus “opportunistic”
- IPv6 and IPv4
- How to build protocols to be NIC offload friendly

Basic Offloads

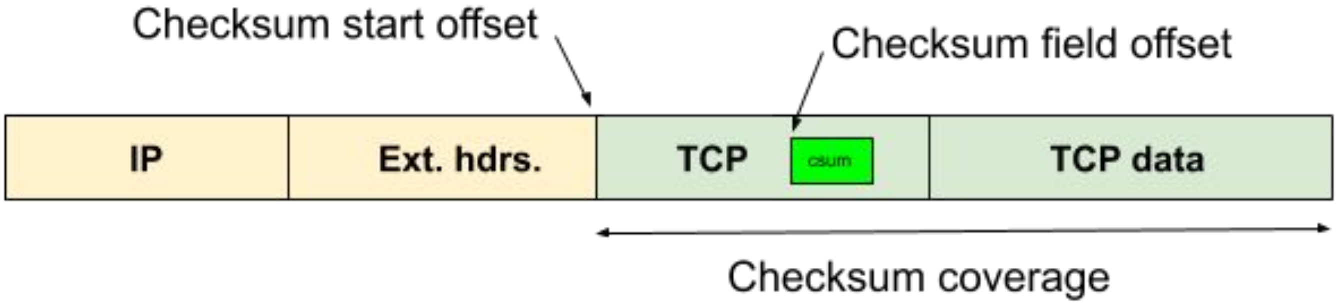
- Checksum offload
- Segmentation offload
- Multi-queue and packet steering

Checksum Offload

- TCP, UDP, GRE, etc...
- NIC offload calculation over data
- Checksum offload is ubiquitous
- Encapsulation allows multiple checksums in same packet

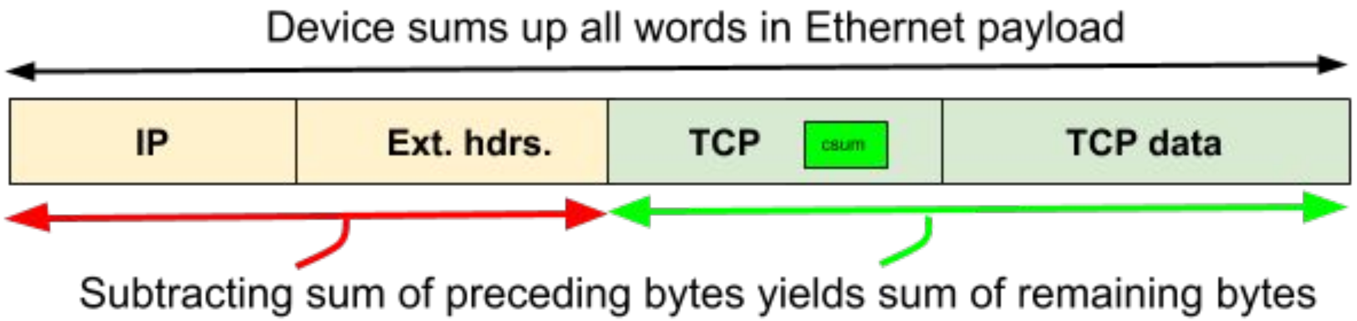
TX Checksum Offload

- Device parses tranports and set checksum
  - Device parse packets and set TCP or UDP checksum
- Instruct device where to start and write checksum
  - Init csum field, indicate start offset and offset to write csum
  - Generic method



RX Checksum Offload

- Checksum unnecessary
  - Device parses packet and verifies UDP or TCP checksum
- Checksum complete
  - Device return 1’s complement sum across words in the packet
  - Generic method

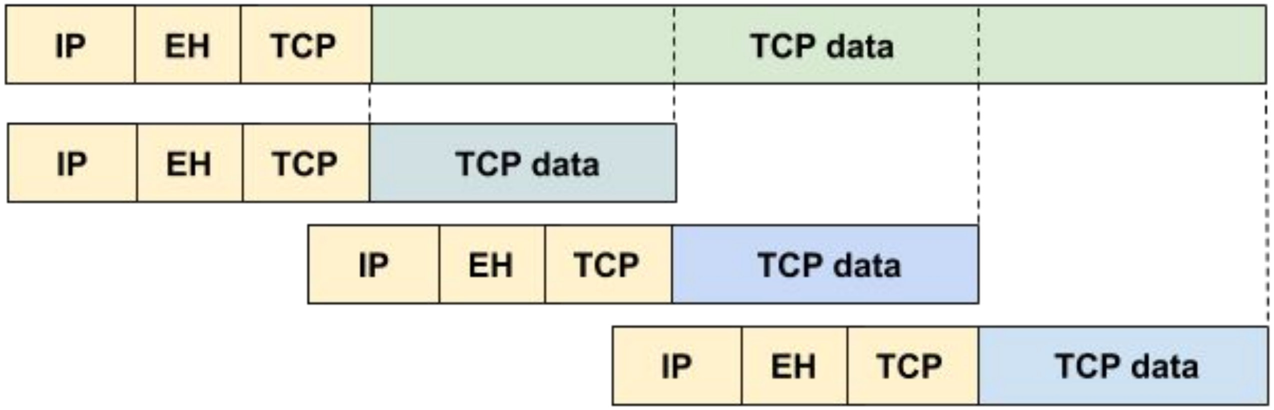


Segmentation Offload

- Stack operates more efficiently on large packets
- Combines with checksum offload to minimize header processing and per packet overhead

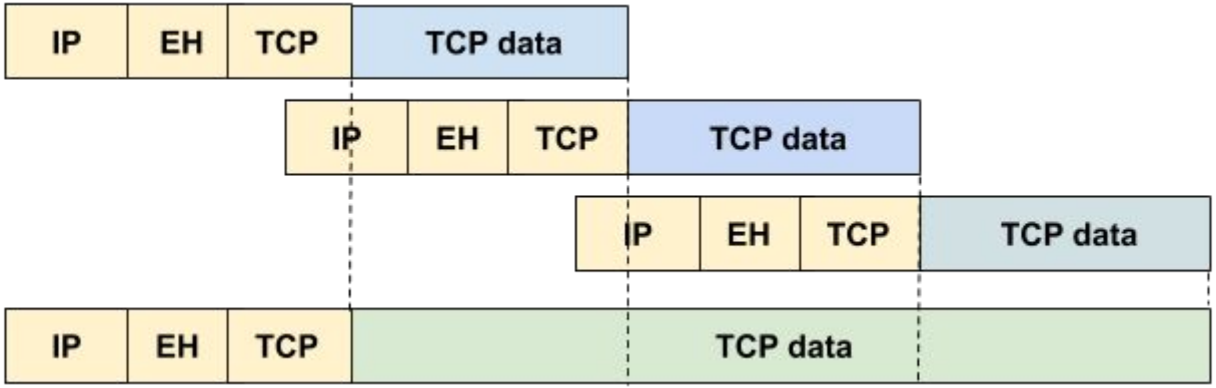
Transmit Segmentation Offload

- Split big packet into smaller one low in the stack
- GSO, Generic Segmentation Offload: SW variants
- LSO: HW variant



Receive Segmentation Offload

- Coalesce small packets into bigger ones low in stack
- Generic Receive Offload, GRO: SW variant
- Large Receive Offload, LRO: HW variant
- Difficult to make protocol agnostic!

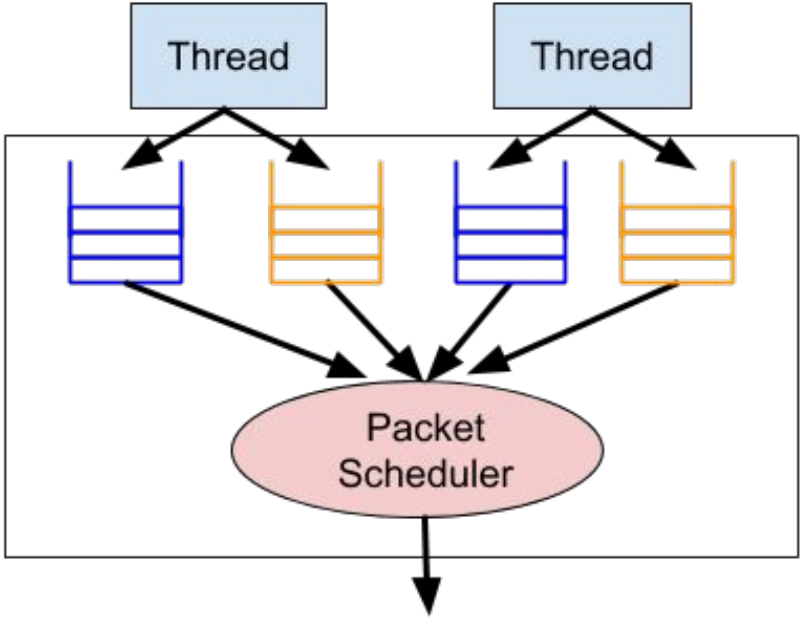


Multi-Queue

- Multiple queues exposed by NIC
- Queues processed by CPUs
- Queues can be accessed and processed in parallel, technique for load balancing
- Queues can also have different properties, e.g. priority
- Avoid OOO packets, maintain flow to queue affinity

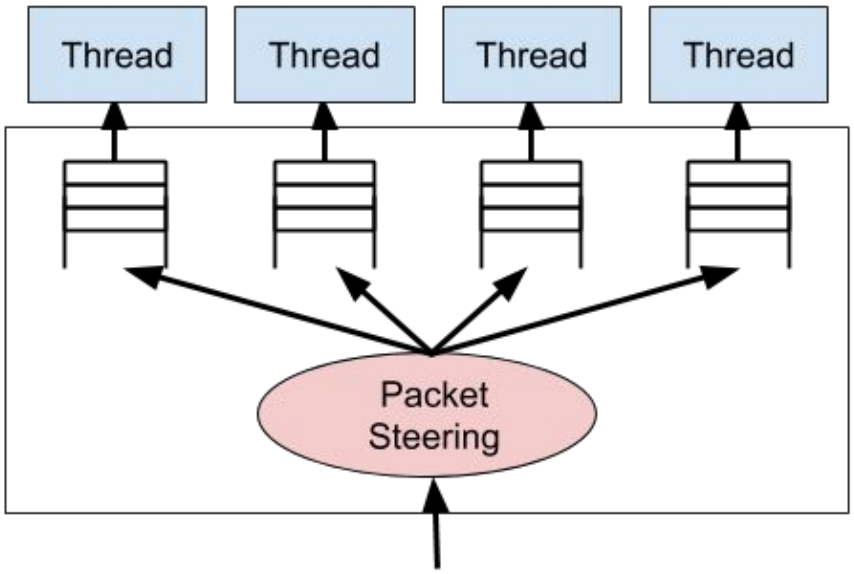
Transmit Queue Selection

- XPS, Transmit packet steering
    - Send packets on queue associated with CPU or thread
- Driver selects queue
    - Device driver operation
    - **ndo_select_queue** in Linux
    - Arbitrary properties (e.g. priority)



Receive Packet Steering

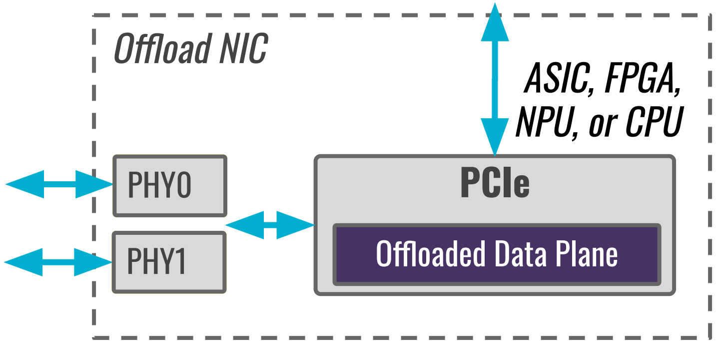
- Receive Packet Steering
  - Steer to queue based on hash
  - RPS is SW variant
  - RSS, Receive Side Scaling, is HW
- Receive Flow Steering
  - Flow to queue association
  - RFS is SW variant
  - aRFS, accelerated RFS, is HW variant



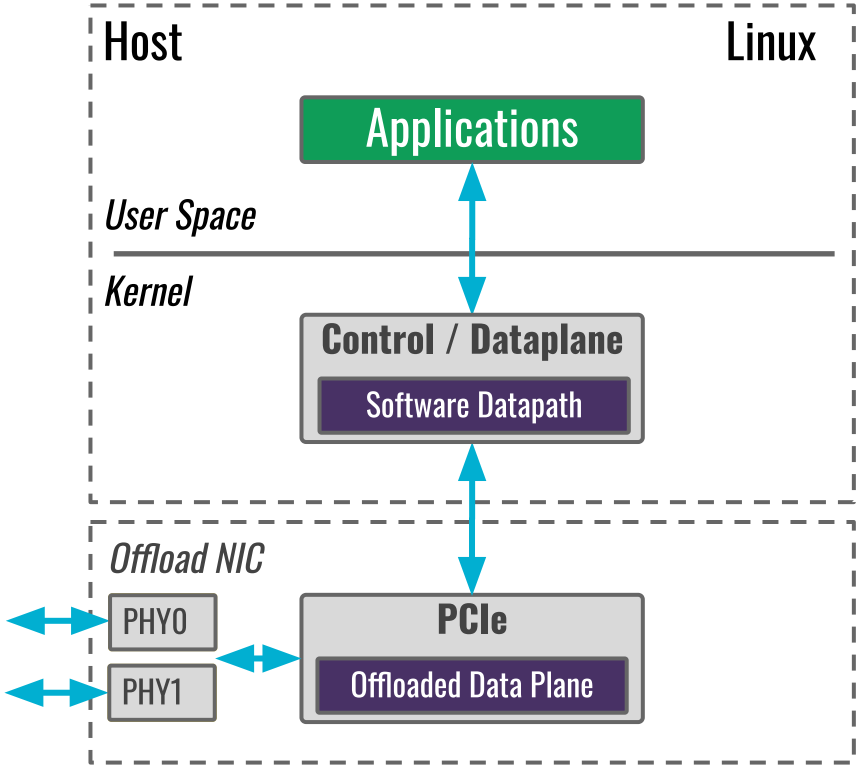
Data Plane in Hardware

Data Plane in Hardware

- Fixed or minimally configurable pipeline
  - ASIC with TCAM tables used for configuring pipeline
- Programmable Pipeline
  - Network Processing Unit/Network Flow Processor
    - Multi-threaded execution environment for data plane programs
  - FPGA
    - Gate-Level Programmable
  - General Purpose Processor
    - CPU Complex separate from host



Data Plane Acceleration

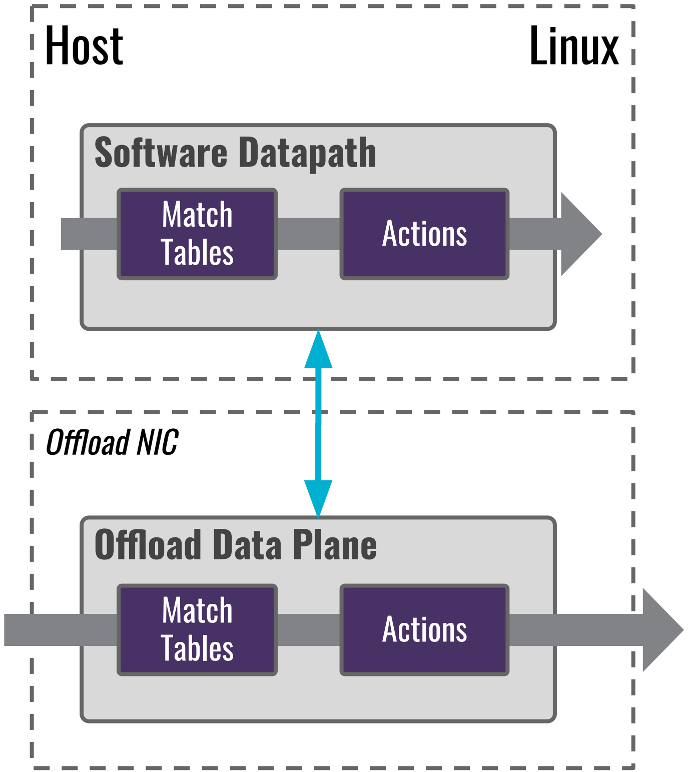


- Control plane stays in host software stack
- Offload data plane
- Hardware Fallback/ Assist datapath in host software stack
- Host software stack implements features of offload data plane

Data Plane Acceleration

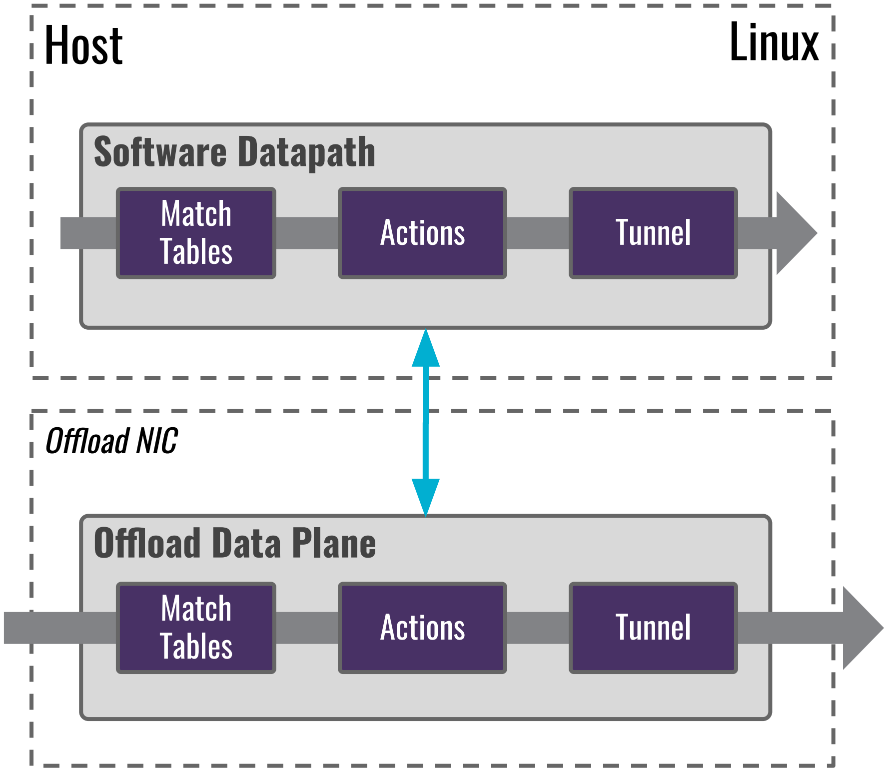
- Match/Action
- Forwarding
- QoS
- TLS and IPsec

Match/Action



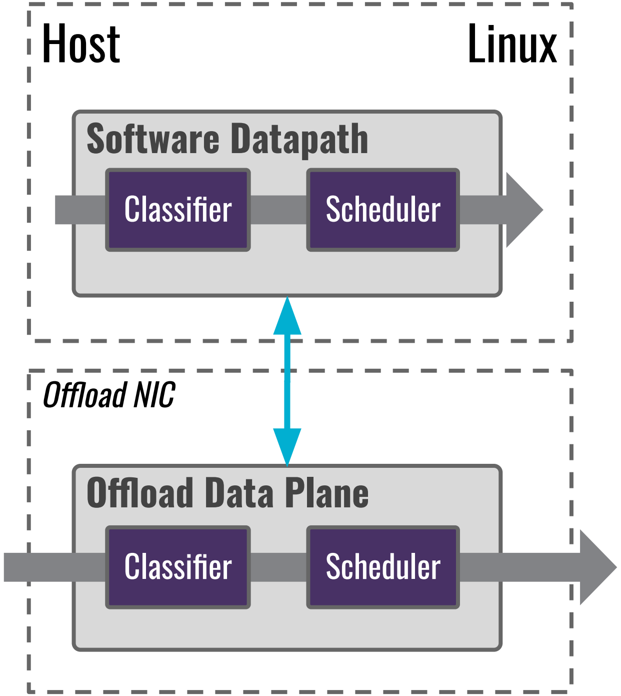
- Match packet based on headers and metadata
  - e.g: input-device + 5-tuple
- Execute actions based on match
  - Forward / Mirror
  - Drop
  - Packet/metadata modification
- Stateful actions
  - Policing
  - Connection tracking

Forwarding



- L2 -> Ln
- Between physical and logical devices
- HW datapath misses can fall back to host
- Optional tunnel encap/decap
  - VXLAN, GRE, Geneve, …
- And tagging: VLAN, MPLS

QoS

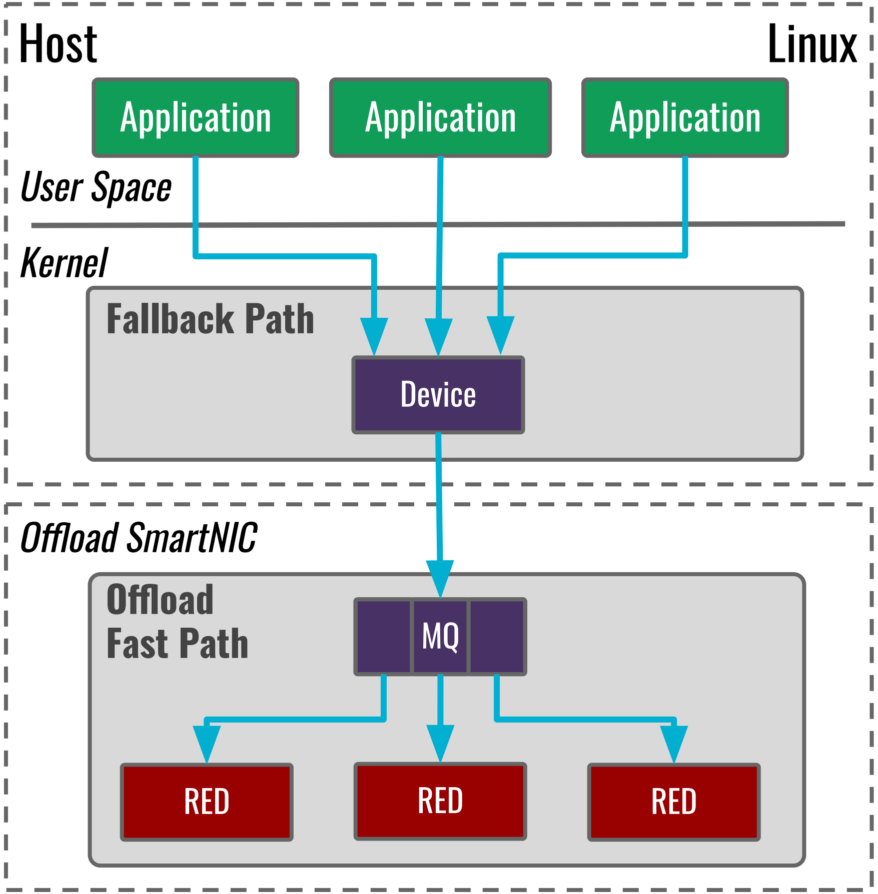


Egress QoS

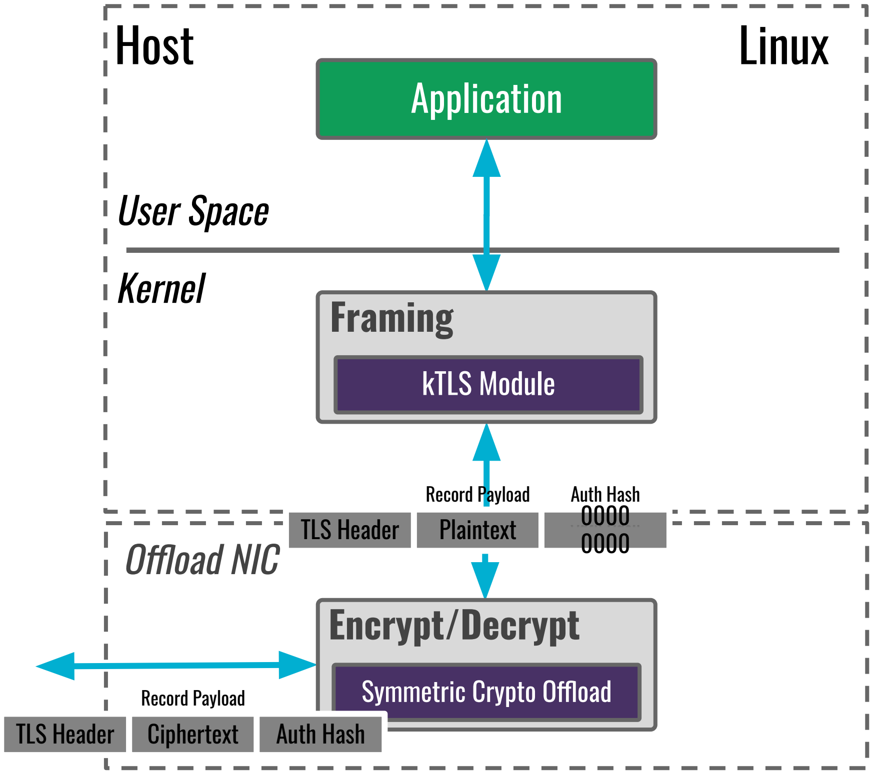
- Ingress
  - No queue
  - Police/Meter/Filter
- Egress
  - Classifier selects priority
  - Scheduler
    - Priority Scheduler, f.e. 802.1p
    - Deficit round robin
    - TSN
    - Shaping: DCB, …

MQ + RED Offload

- Per-device RED in HW
- May ECN mark or drop packets



TLS Acceleration



Established TLS be passed to kTLS

TX Path:

- NIC driver marks packets for crypto offload based on packet socket
- NIC performs encrypt and TX

RX Path:

- NIC performs decrypt and auth
- Notifies kTLS of queued data
- kTLS skips decrypt of plaintext
- Handle Out-Of-Order

IPsec Acceleration

Crypto Offload

- HW: Encrypt/Decrypt/Integrity/LSO/Checksum
- Kernel: Padding/Anti-replay/Counters/Security Policy DB
- User-Space: IKE

Full Offload

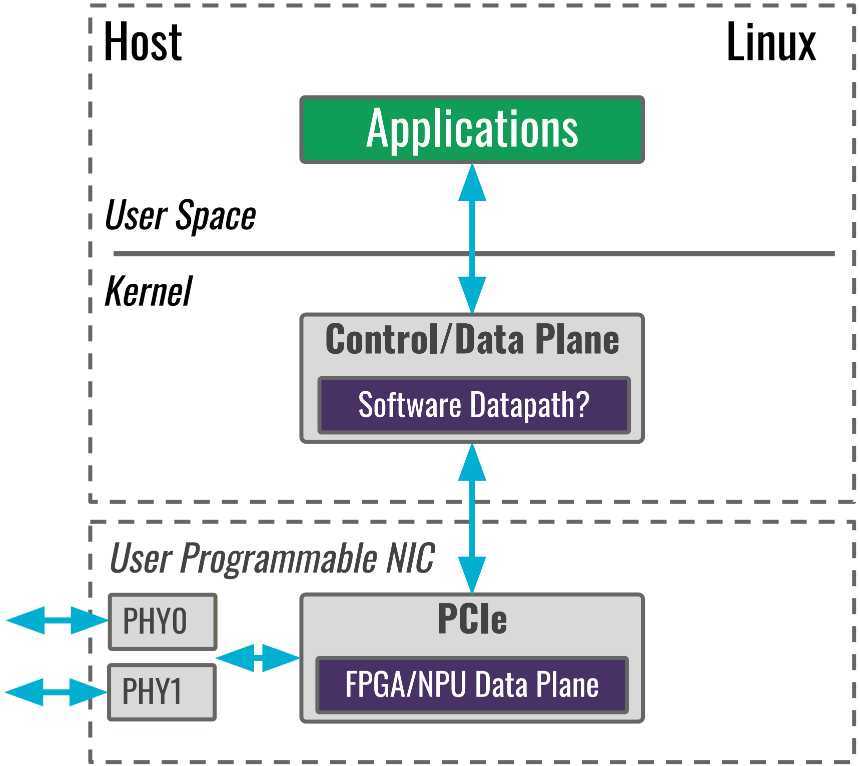
- HW: Replay/Encap/Decap/SPD/LSO/Checksum/LRO
- Kernel: IP fragmentation/Counters/Configuration
- User-Space: IKE

Programmability

Programmability

- Facilitates rapid protocol development
- Quickly fix bugs and security problems
- Two main types used today:
  - FPGA/NPU
  - General Purpose Processors
- Emerging trend: *What is niche today can be broad tomorrow*
  - IETF 104 "Forwarding Plane Realities"

Programmability with FPGA or NPU

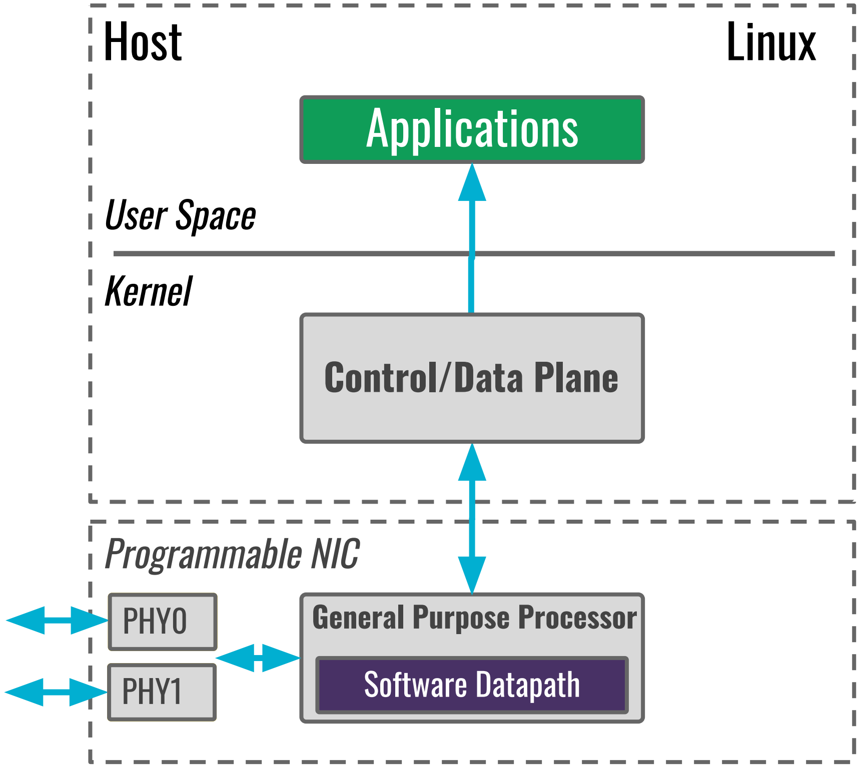


- Control plane stays in host
- Flexible offload data plane controlled through kernel or user space
- Data plane could be expressed by P4, eBPF, NPL, or other native instruction set
- Dynamically programmed

General Purpose Processor

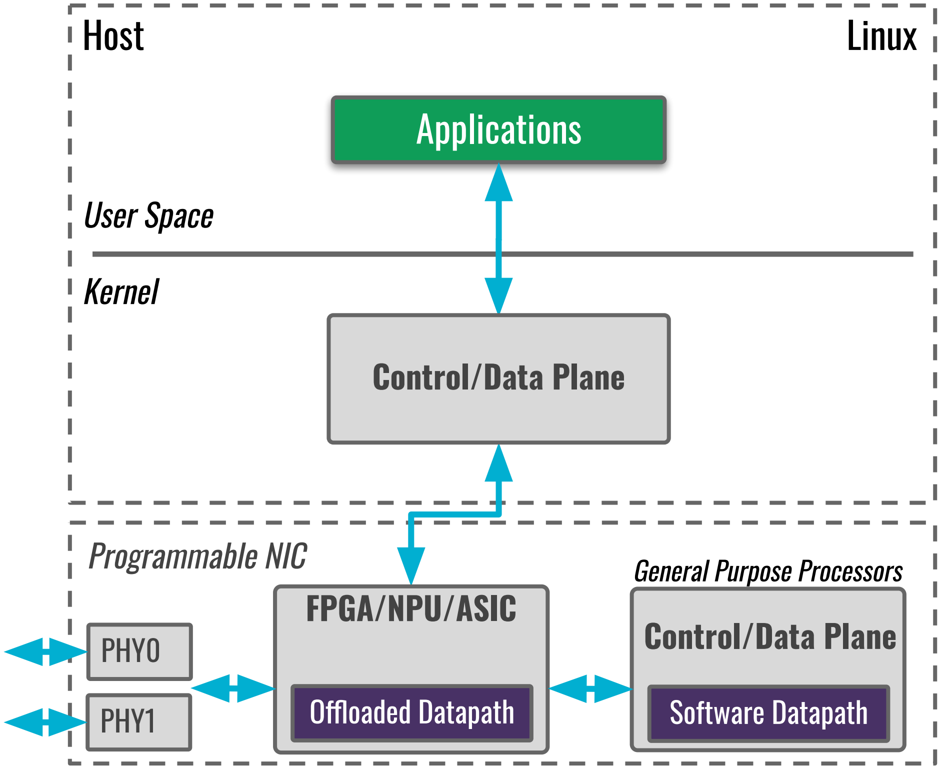
- Move host software stack down to the NIC
- Dataplane offload to general purpose processor on NIC
- Control plane offload
  - Useful in bare metal or multi-tenant deployments
  - Network admin can control server networking
- No host resources consumed forwarding network traffic

Programmable NIC with General Purpose Processor



- Capable of running complete Operating System
- Forwarding functionality moved completely away from server cores down to NIC

Programmable NIC with General Purpose Processor



- Programmable NICs also have offload-capable devices

Conclusion and Futures

- Networking trends
  - Insatiable need for more bandwidth and lower latency
  - Deployment of forward looking IETF protocols
- NICs work with hosts to make this happen
  - Offloads will be relevant for foreseeable future
  - Programmability and flexibility spur innovation

Thank You!