School of Engineering & Applied Science

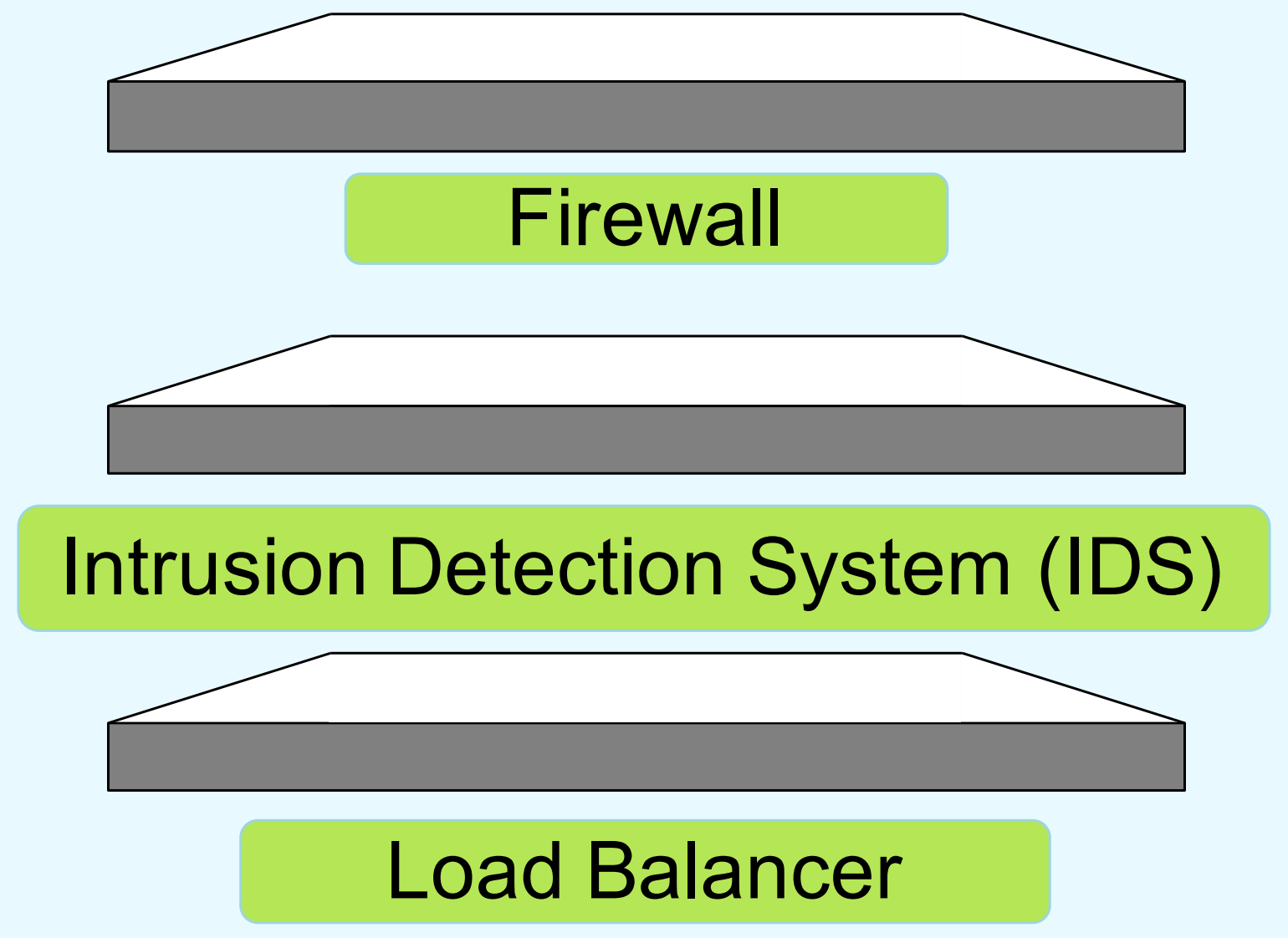
THE GEORGE WASHINGTON UNIVERSITY

openNetVM: Bringing Elasticity to Enterprise Networks using Network Function Virtualization on Commodity Hardware

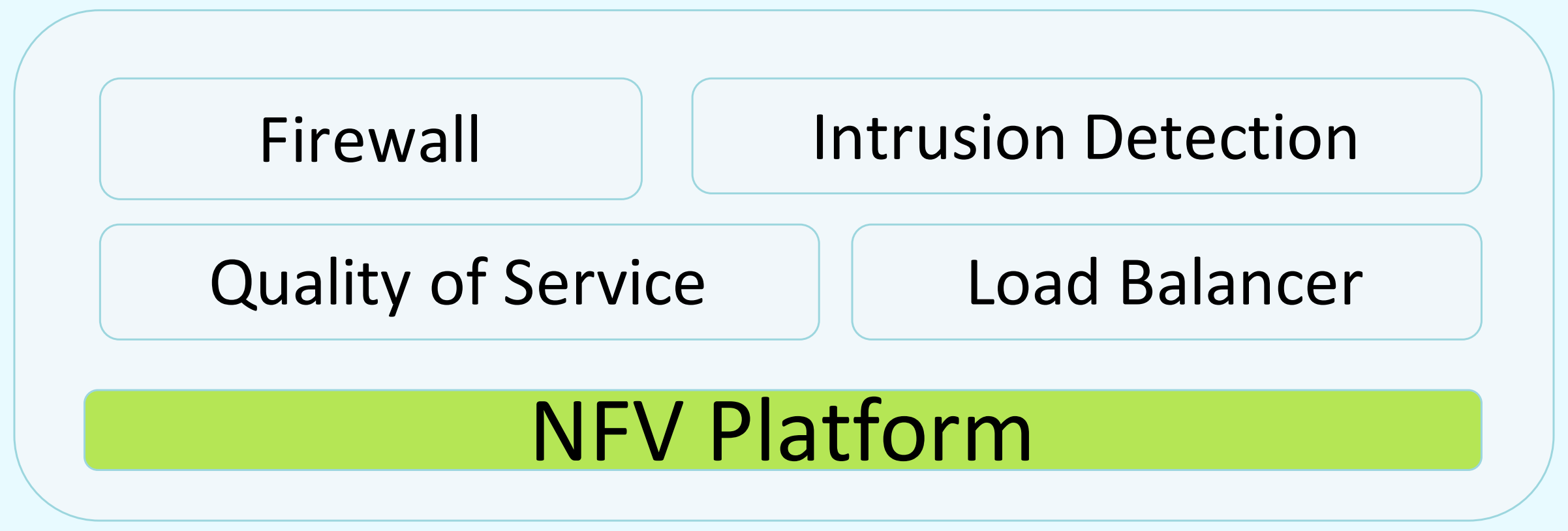
Neel Shah, Philip Lopreiato, Warren Smith, Wei Zhang, and Timothy Wood
The George Washington University

Abstract

- Traditionally, networks are comprised of individual **hardware components** called network functions:


Firewall
Intrusion Detection System (IDS)
Load Balancer

- This model is very **expensive** and **inflexible**
- Trends in networking produced network function virtualization (NFV)
  - Cost effectiveness of software
  - Flexibility of software
  - All network functions on one host


Firewall
Intrusion Detection
Quality of Service
Load Balancer
NFV Platform

Challenges

- Modern NFV technology still does not provide an elastic framework
  - Adding new NF requires network downtime
- Modern NFV technology is not able to perform at the same line rates as hardware networks

openNetVM

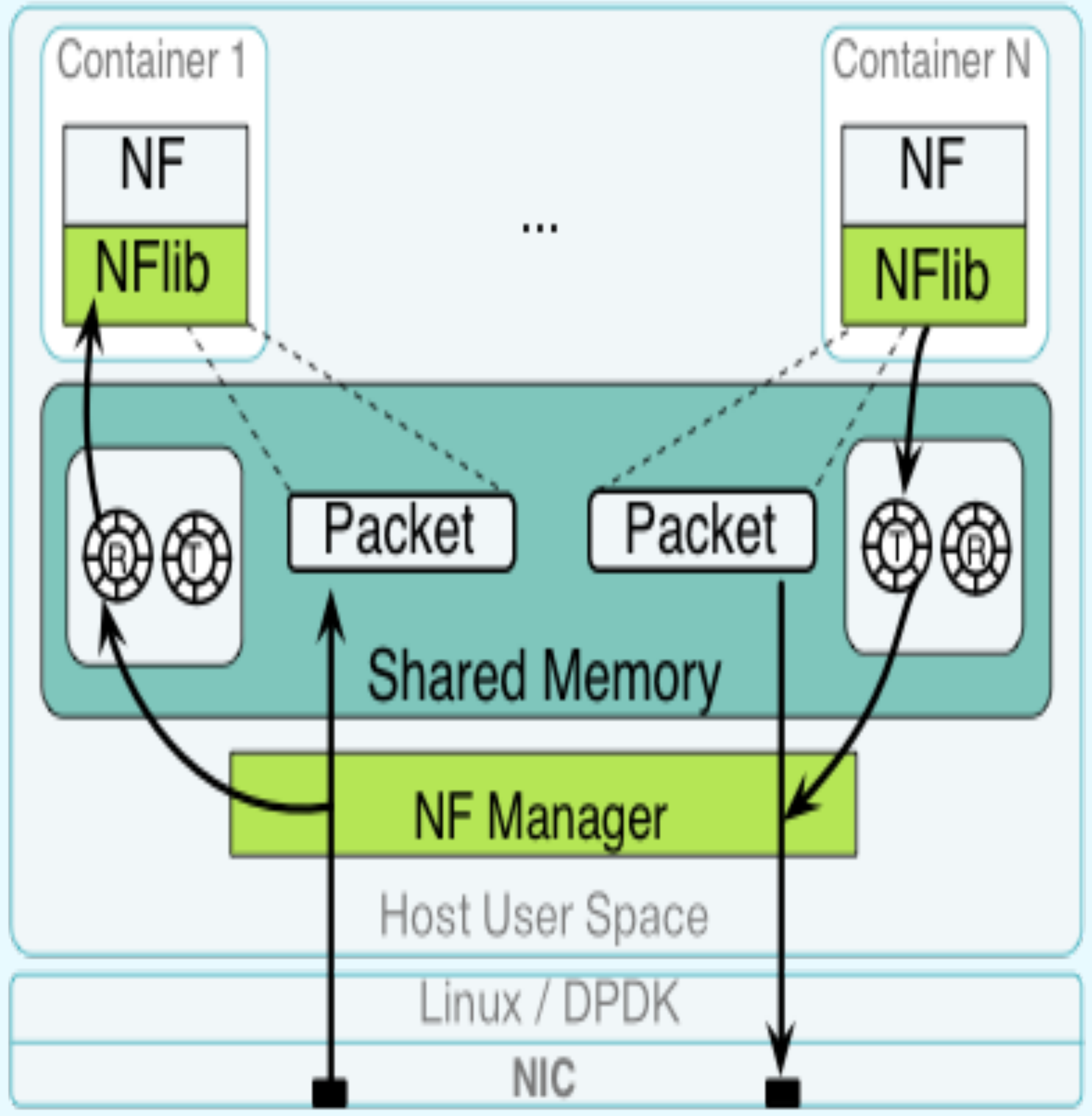
Container based NFs: Ease of user-space process management brought to networking

NF Manager: Orchestrates traffic flow between various NFs to bring elasticity

Zero-Copy IO: Packets DMA'd into shared memory granting NFs direct access to data without copies

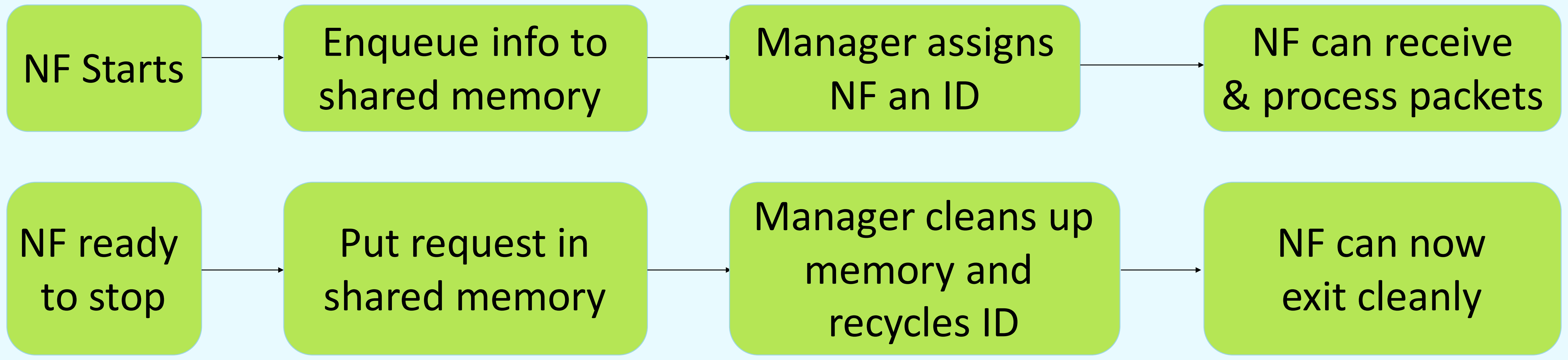
NUMA-Aware: Maximizes performance by ensuring data in memory is local to a thread's CPU Socket

Interrupt-Free: DPDK's poll mode driver allows non-traditional network to process incoming traffic at **10Gbps and beyond**



Dynamic Manager

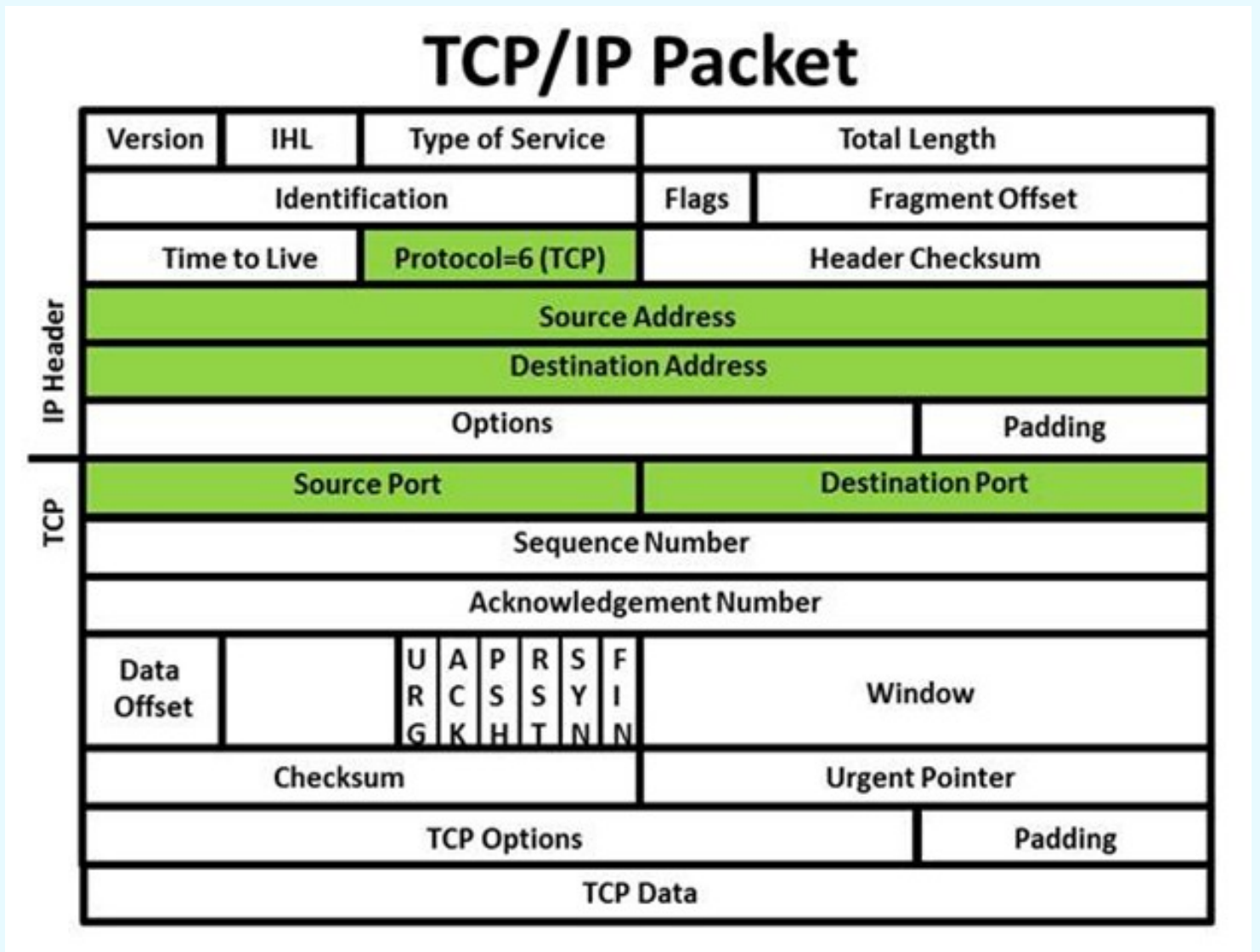
- As networks grow, more middle boxes need to be deployed to scale efficiently
- openNetVM has a dynamic manager which makes networks elastic
  - Aware of all active and newly created NFs
  - Re organizes data structures upon NF creation and destruction
- Dynamic NF start and stop protocols let the size of the network scale in proportion to traffic without downtime
- System can recover from and restart crashed NFs


NF Starts → Enqueue info to shared memory → Manager assigns NF an ID → NF can receive & process packets

NF ready to stop → Put request in shared memory → Manager cleans up memory and recycles ID → NF can now exit cleanly

TCP/IP Library

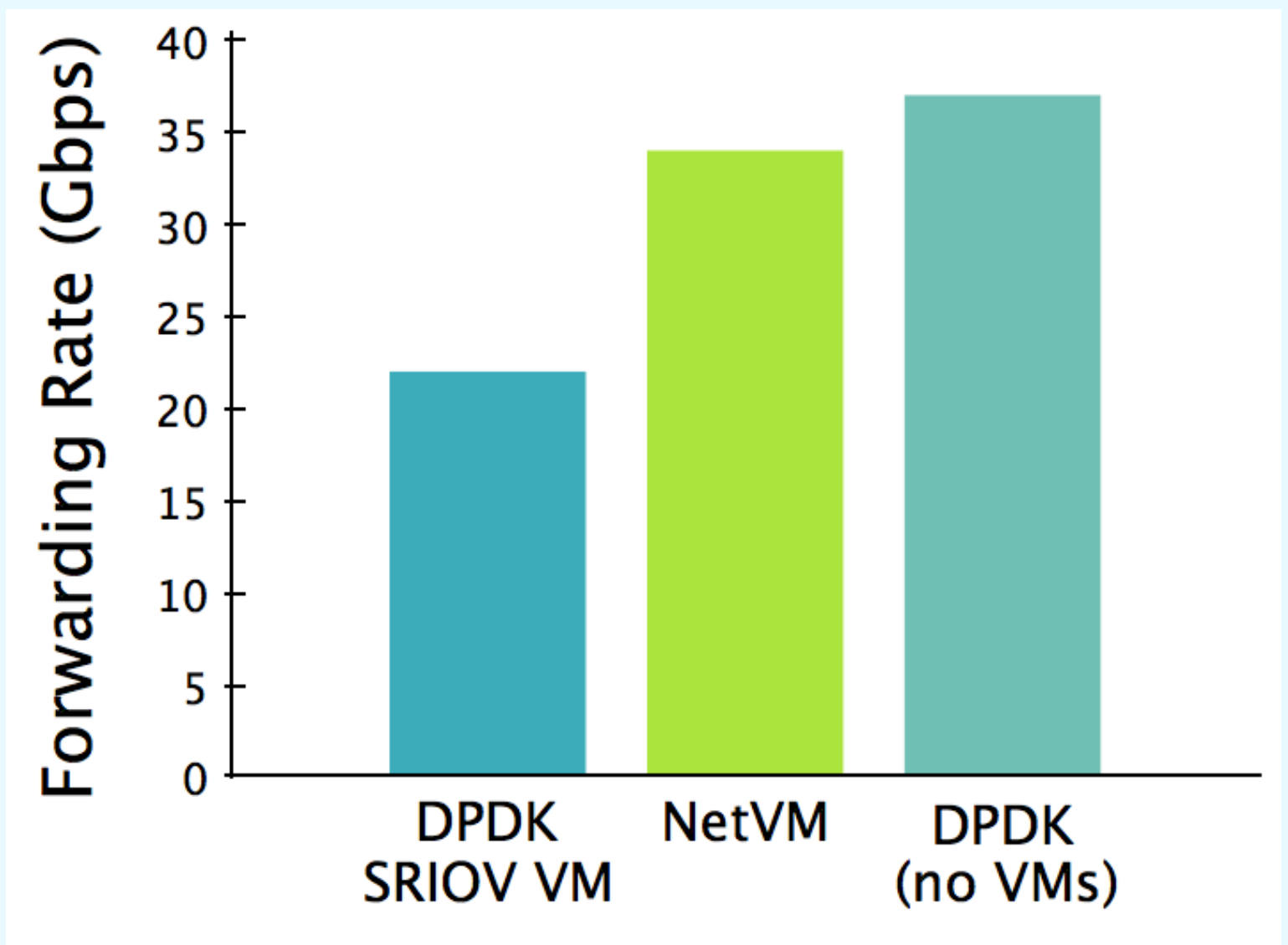
- DPDK strips standard packet headers from traffic since it avoids the kernel
- More complicated network functions (IDS, firewall) need TCP/IP packet headers to perform their tasks
- TCP/IP library exposes the standard headers from the packets


TCP/IP Packet

IP Header
Version | IHL | Type of Service | Total Length
Identification | Flags | Fragment Offset
Time to Live | Protocol=6 (TCP) | Header Checksum
Source Address
Destination Address
Options | Padding

TCP
Source Port | Destination Port
Sequence Number
Acknowledgement Number
Data Offset | | URG | ACK | PSH | RST | SYN | FIN | Window
Checksum | Urgent Pointer
TCP Options | Padding
TCP Data

Results



- Comparing SR-IOV enabled VMs and DPDK against NetVM (our other system that uses VMs for the same goal), we achieve line rates that are faster than SR-IOV VMs but not faster than raw DPDK
- We expect openNetVM to be as fast as raw DPDK or faster than it since containers are much lighter