DPDK Integration within F5 BIG-IP

BRENT BLOOD, SR MANAGER SOFTWARE ENGINEERING

VIJAY MANICKAM, SR SOFTWARE ENGINEER

F5 Company Snapshot

Founded: **1996**

IPO: June **1999**

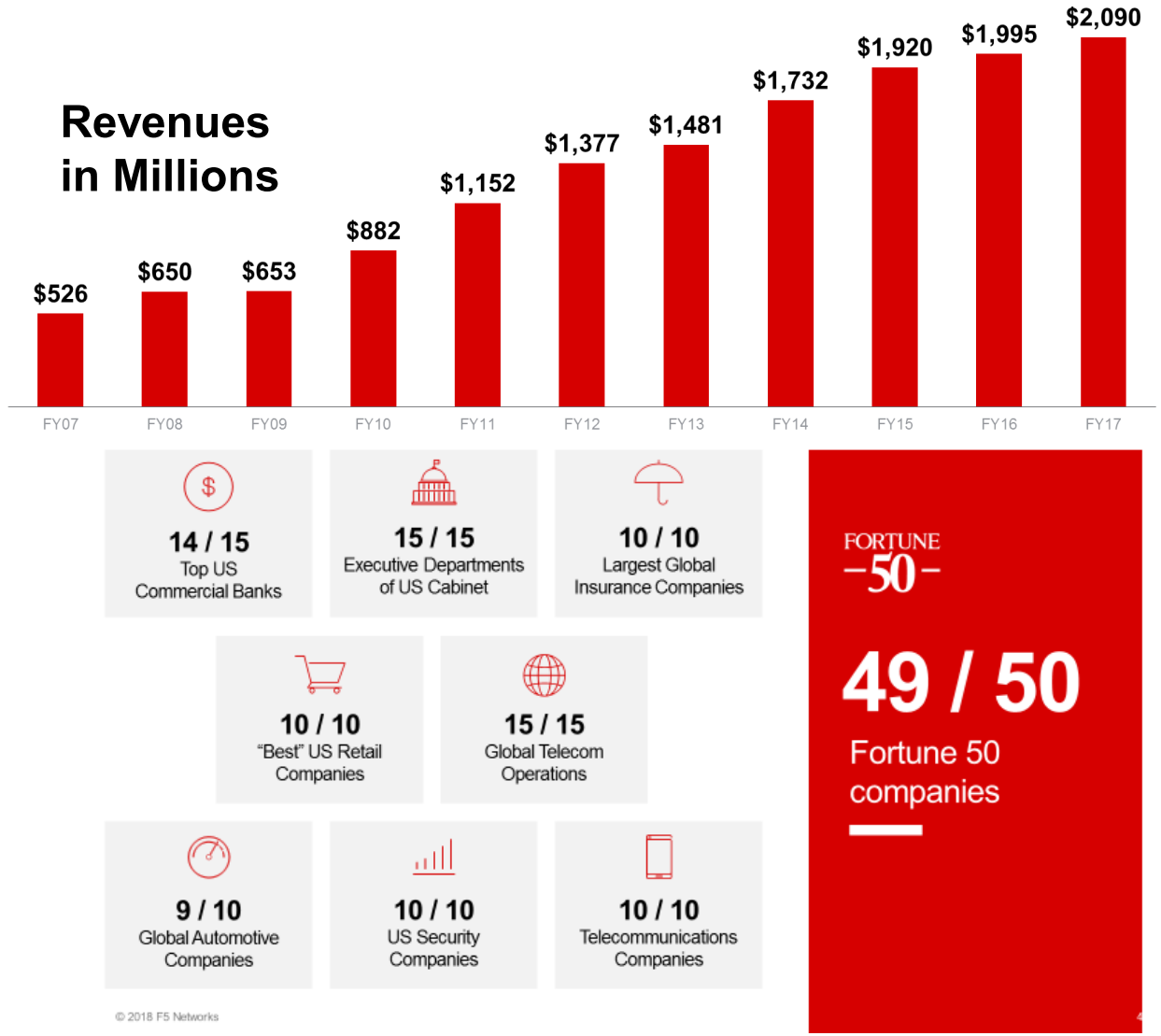
Employees: **4,395**

Headquarters: **Seattle, WA**

President and CEO: **François Locoh-Donou**

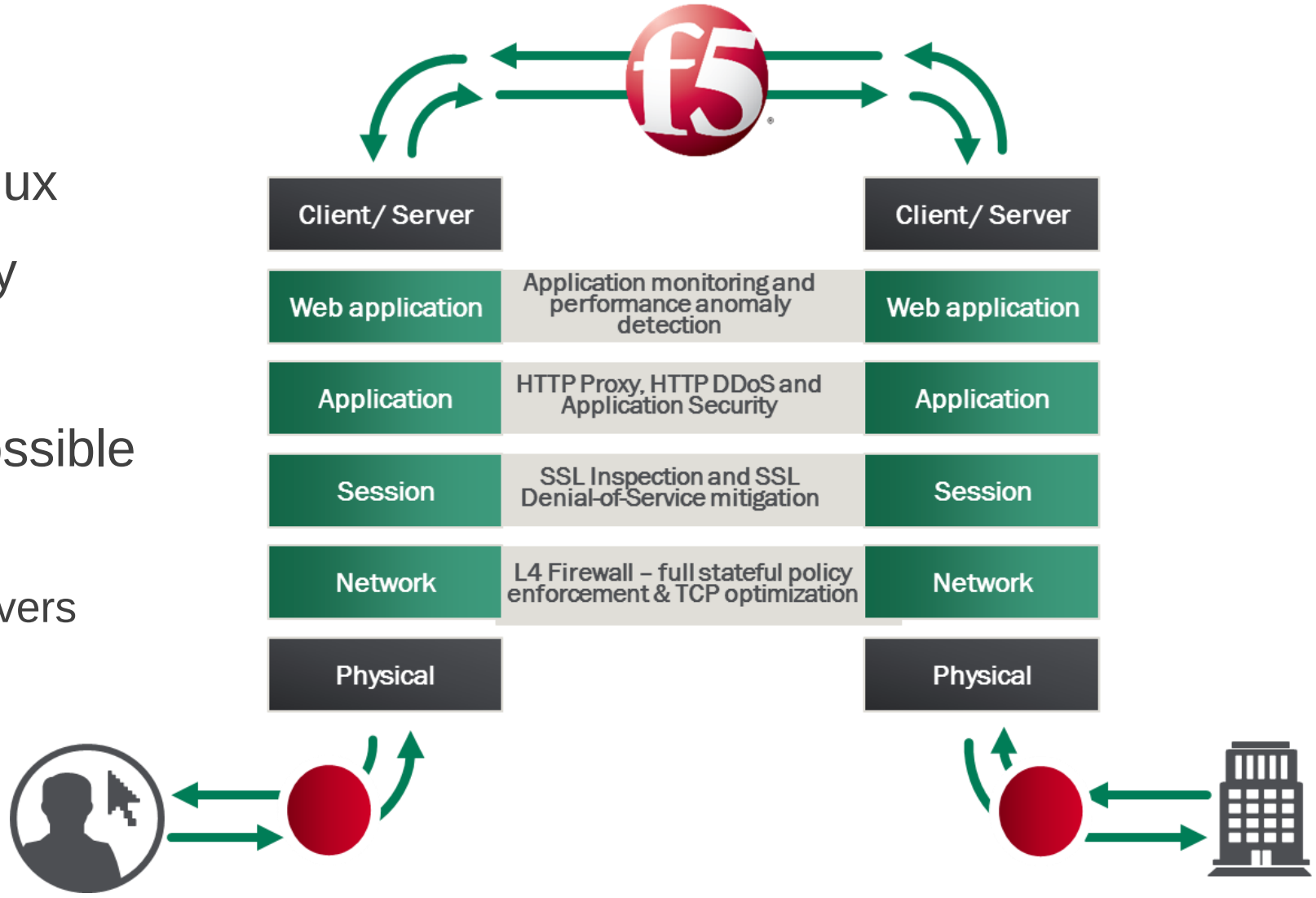
Market symbol: **FFIV (NASDAQ)**

Operations worldwide: **32 countries**



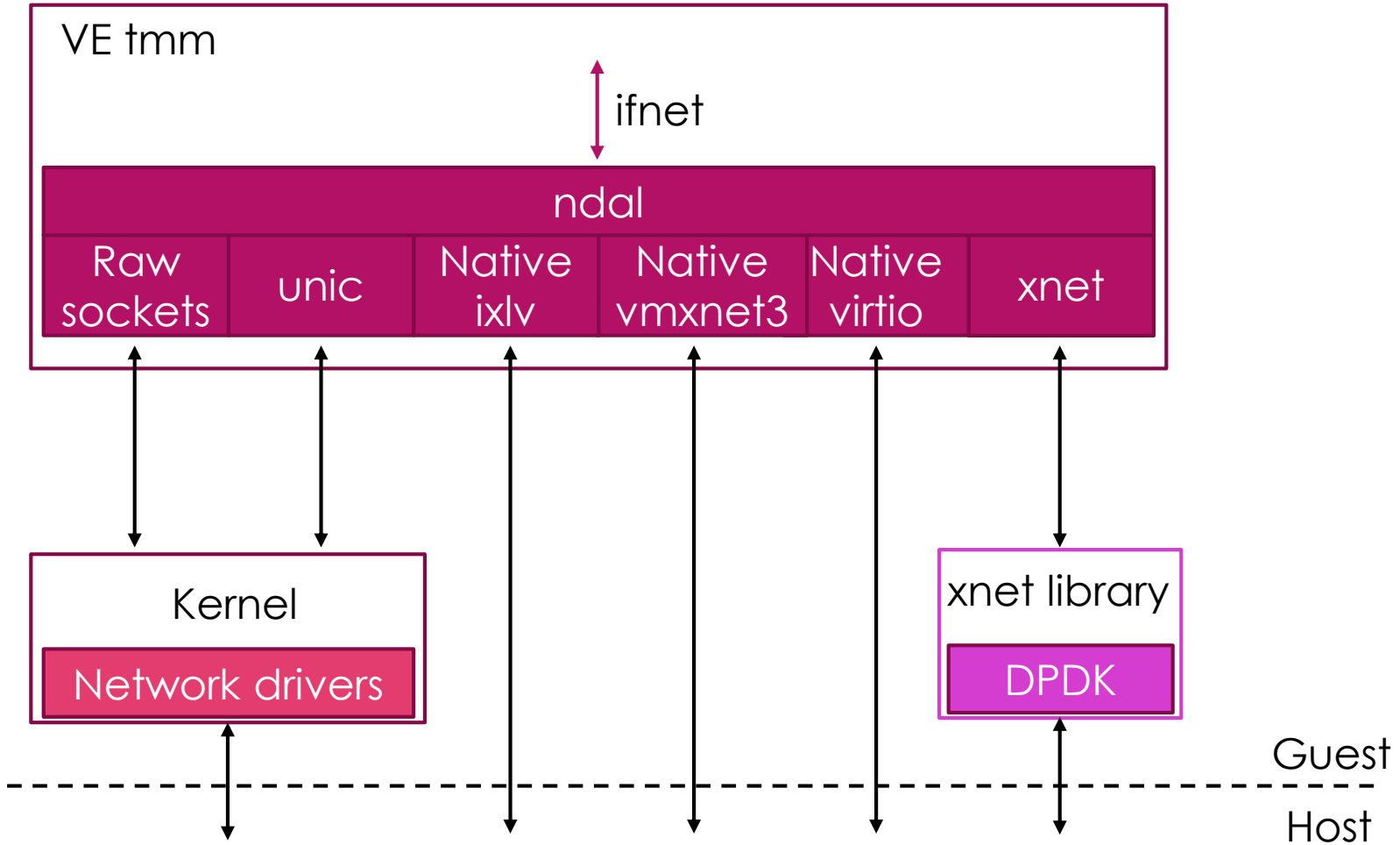
tmm – Traffic Management Microkernel

- Userland process in Linux
- Modular L2-L7 full proxy
- User programmable
- Zero copy whenever possible
- Packet models:
  - Native poll mode PCI drivers
  - Raw socket via kernel
  - DPDK

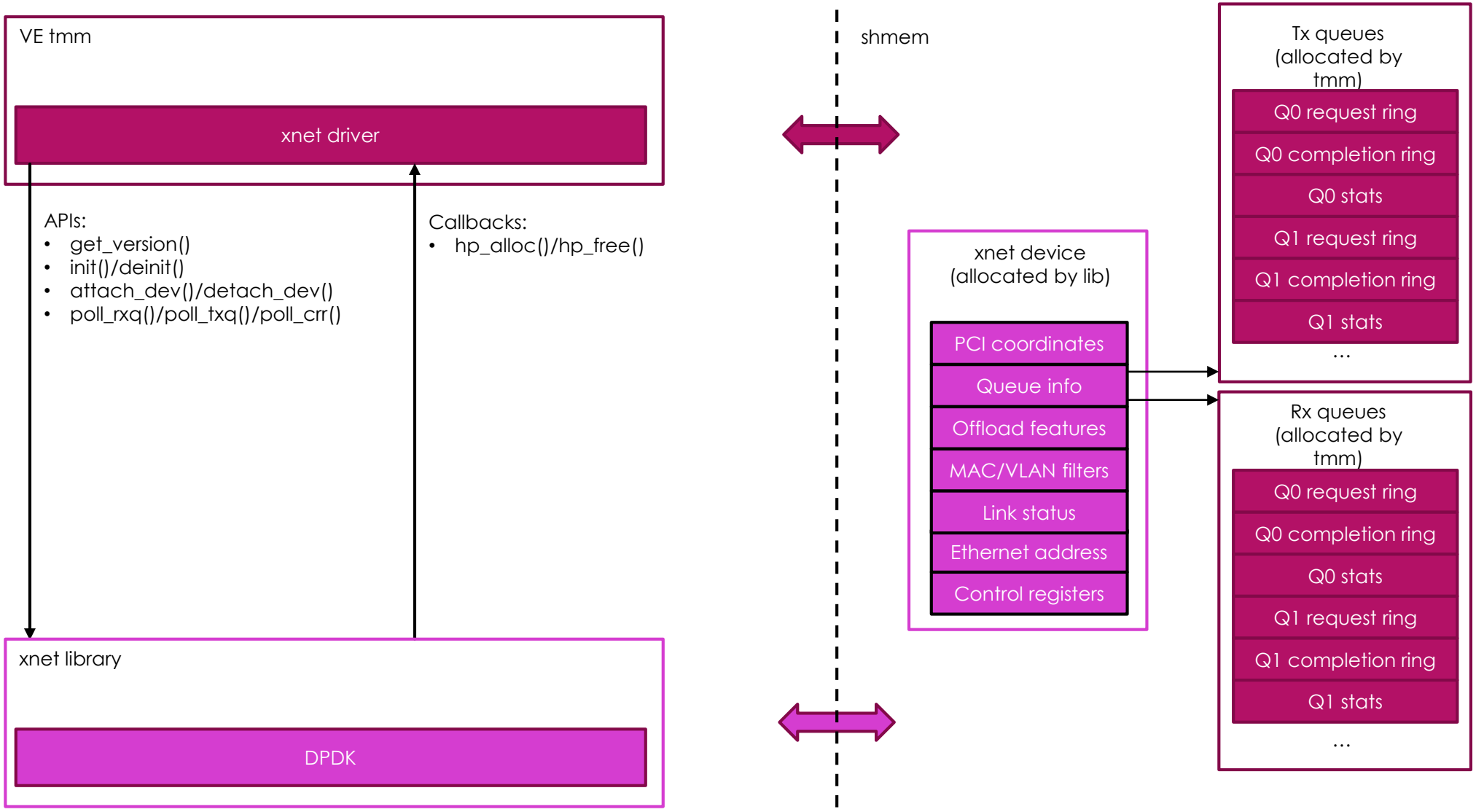


xnet: Integrating DPDK with tmm

- Sits between the tmm and DPDK to isolate changes
- DPDK is not aware of the tmm
- The tmm is not aware of DPDK



A closer look at tmm/xnet/DPDK



Challenge #1: Foreign Memory Import API

- The tmm manages all hugepages in system
- The tmm has its own purpose built memory manager
- DPDK runs within the same process as the tmm
- DPDK assumes it owns all hugepages – but it doesn't / cannot!

- Solution: Tmm will feed hugepages to DPDK before doing rte_eal_init() providing it with the list the hugepages DPDK can own

Challenge #2: Zero-copy retaining mbuf and xfrag

- DPDK uses mbuf structures but tmm uses xfrags
- Both contain effectively the same fields because they do the same thing
- We could convert one to the other – but this defeats zero copy and hurts performance
- We don’t want to change all of tmm to use mbufs – but don’t want to maintain a patch against DPDK to do the opposite
- Solution: Tweak DPDK’s external mempool handler to attach xfrags to mbuf’s payload and translate mbuf header to xfrag header – Rx and xfrag header to mbuf header - Tx

Challenge #3: Contiguous Hugepages

- DPDK normally acquires hugepages from Linux
- It then mmap()s them into a contiguous address space
- When inside of tmm, we skip that
- tmm then sets aside some hugepages, sorts the physical pages, and remaps to a contiguous virtual space

Challenge #4: mbuf cache

- Freed Tx xfrags need to be freed to TMM ASAP.
- So we had to TURN OFF the mbuf cache
- Performance degrades with more rx/tx queues like 16 each
- Solution: Implement an mbuf cache in xnet_lib layer
- Proposal: DPDK to support freeing of mbuf payload while still maintaining the mbufs in the cache.

Challenge #5: Allocating and Freeing Buffers

DPDK
DATA PLANE DEVELOPMENT KIT

- DPDK and the tmm have incompatible models for allocating and freeing buffers:
  - DPDK wants to allocate a bunch of buffers up front
  - DPDK wants to free buffers in bulk
  - The tmm expects the opposite of these
- Solution: use xnet library to coalesce behaviors to insulate tmm and DPDK from each other

Challenge #6: DPDK drivers with dependencies

- Some NICs can be driven by DPDK without external dependencies
  - Easy to enable for evaluation
  - Easy to integrate into our build system
  - Easy to upgrade DPDK library
  - Easy to track errata & vulnerabilities, minimizes our surface
  - Simple licensing **(no need to involve lawyers)**
- Others require external kernel modules and libraries
  - Opposite of all above points
  - Dealing with this is more work than writing our own driver in some cases, and it doesn’t even perform as well!

Thank You!

Q & A