

Linux eBPF XDP at 400GbE Engineering Handbook

Driver Hooks, SmartNIC Hardware Offload, AF_XDP Zero-Copy, and Production Kernel Tuning

EXECUTIVE ABSTRACT & SUMMARY FINDINGS: As enterprise and cloud backbones transition to 100GbE and 400GbE, standard Linux networking stacks encounter insurmountable packet allocation overhead in the `sk_buff` subsystem. While DPDK achieved kernel bypass, it monopolizes CPU cores in 100% busy-polling loops. eBPF eXpress Data Path (XDP) provides in-kernel programmable packet processing directly in the NIC driver at line rate. This handbook examines XDP driver hooks, AF_XDP zero-copy socket ring buffers, tail calls, and SmartNIC hardware offload physics.

Network Framework	Packet Rate (Mpps/core)	CPU Utilization	Linux Tooling Support
Standard Linux <code>sk_buff</code>	1.8 - 2.4 Mpps	100% (High SoftIRQ)	Full Native Support
Linux XDP (Driver Mode)	24.5 - 28.2 Mpps	100% (Linear Scale)	Full Native Support
DPDK (Kernel Bypass)	32.0 - 36.5 Mpps	100% (Dedicated Poll)	Zero (Kernel Blind)
XDP Hardware Offload	Line Rate (400Gbps)	0% (NIC NPU/FPGA)	Full Native Support

1. The `sk_buff` Allocation Bottleneck at Extreme Line Rates

At 400 Gigabit Ethernet line rate, receiving minimum-sized 64-byte Ethernet frames delivers approximately 595 million packets per second. This yields a microscopic budget of barely 1.68 nanoseconds per packet. In the standard Linux network stack, the NIC interrupt handler allocates a 240-byte `sk_buff` struct for every incoming frame, performs atomic refcounting, and traverses the entire IP routing, conntrack, and Netfilter firewall table.

Because `sk_buff` metadata allocation and cache eviction alone consume 45 to 60 nanoseconds, standard Linux sockets collapse under heavy packet load, dropping frames and saturating multiple CPU cores in `‘ksoftirqd’` spinning loops.

2. The Three Operating Modes of eXpress Data Path (XDP)

eBPF eXpress Data Path (XDP) executes sandboxed verification bytecode at the lowest possible layer in the network subsystem, operating in three distinct modes:

- (1) Generic XDP (`xdpgeneric`): Runs after `sk_buff` allocation, primarily used for testing and validation on unsupported NICs.
- (2) Native/Driver XDP (`xdpdrv`): Runs directly inside the network driver's RX ring buffer before `sk_buff` allocation. The driver passes raw DMA packet memory (`xdp_buff`) to the eBPF program, achieving 24+ million packets per second per CPU core.
- (3) Hardware Offload (`xdpoffload`): The eBPF bytecode is JIT-compiled into native FPGA or NPU microcode and executed directly on a SmartNIC (such as AMD Pensando or NVIDIA BlueField), filtering packets at full 400Gbps wire speed with 0% host CPU load.

3. AF_XDP Zero-Copy (UMEM) User-Space Sockets

When user-space applications (such as high-frequency trading gateways or load balancers) require raw packet access, the XDP program executes an `XDP_REDIRECT` action into an AF_XDP socket (XSK).

AF_XDP utilizes a shared memory pool (UMEM) mapped into both the kernel driver and user-space virtual memory via two pairs of lockless ring buffers: the Fill Ring and Rx Ring for ingress, and the Tx Ring and Completion Ring for egress. DMA transfers write packet payloads directly into UMEM buffers without memory copies, achieving DPDK-equivalent sub-microsecond latency while maintaining standard Linux namespace isolation and routing semantics.

4. Production Tuning: RSS, NAPI, and NUMA Affinity

Saturating 400GbE line rates requires meticulous hardware-level alignment. Network interface Receive Side Scaling (RSS) queues must be mapped 1:1 to dedicated CPU cores pinned to the same NUMA node as the PCIe root complex. Setting `ethtool -C eth0 adaptive-rx off rx-usecs 16` optimizes interrupt coalescing, while enabling `XDP_FLAGS_UPDATE_IF_NOEXIST` guarantees atomic program swapping during production firmware upgrades.

CRITICAL ENGINEERING TAKEAWAYS & DEPLOYMENT DIRECTIVES:

1. Standard Linux `sk_buff` allocation consumes 50ns per packet, failing completely at 100G/400G line rates.
2. Native XDP executes eBPF filters directly in the driver's RX ring before `sk_buff` creation (24M+ pps/core).
3. Hardware-offloaded XDP JIT-compiles eBPF into SmartNIC hardware, filtering 400GbE line rate at 0% CPU overhead.
4. AF_XDP zero-copy UMEM ring buffers match DPDK throughput while preserving Linux routing, tools, and security models.