A Look at Intel’s Dataplane Development Kit

Dominik Scholz
Chair for Network Architectures and Services
Department for Computer Science
Technische Universität München

June 13, 2014
Outline

1. Packet Processing using Commodity Hardware
2. Intel’s Dataplane Development Kit
3. Comparison: Intel’s DPDK, netmap, PF_RING DNA
Motivation

Why use commodity hardware and do packet processing in software?

Advantages:

- Flexibility: software can be modified
- Increased performance and reduced costs of (multicore) CPU’s and NIC’s over the last years
- Open source

But: existing dataplane software not designed for high-speed packet processing (up to 10 Gbit/s)

→ specialized frameworks implement different techniques to achieve significant performance speed-ups
Use Case: Linux Network Stack

Operating System

Applikation

Routing Table

Memory

Buffer

Ingress Network Board

Egress Network Board

User Mode

Kernel Mode
Performance Limitating Factors

- Bottleneck CPU
- Bottleneck memory:
  1. per packet allocation and deallocation
  2. multiple copy operations per packet
  3. complex *sk_buff* structure
- Parallelism: spinlocks (active waiting)
- Context switches

Conclusion: standard dataplane only for general purpose
Performance Limitating Factors

- Bottleneck CPU
- Bottleneck memory:
  1. per packet allocation and deallocation
  2. multiple copy operations per packet
  3. complex sk_buff structure

- Parallelism: spinlocks (active waiting)
- Context switches

```
struct sk_buff {
    [...]}

/* Transport layer header */
union
{
    struct tcphdr *th;
    struct udphdr *uh;
    struct icmphpdr *icmph;
    struct igmphdr *igmph;
    struct iphdr *iph;
    struct spxhrdr *spxh;
    unsigned char *raw;
} nh;

/* Network layer header */
union
{
    struct iphdr *iph;
    struct ipv6hdr *ipv6h;
    struct arphdr *arph;
    struct ipxhdr *ipxh;
    unsigned char *raw;
} nh;

[...]
```

Conclusion: standard dataplane only for general purpose
Bottleneck CPU

Bottleneck memory:
1. per packet allocation and deallocation
2. multiple copy operations per packet
3. complex `sk_buff` structure

Parallelism: spinlocks (active waiting)
Context switches

Conclusion: standard dataplane only for general purpose
Outline

1. Packet Processing using Commodity Hardware
2. Intel’s Dataplane Development Kit
3. Comparison: Intel’s DPDK, netmap, PF_RING DNA
Intel DPDK

Set of libraries to accelerate basic dataplane functions

- Released in 2012
- Completely replaces the network stack
- Intel architecture-based: supporting Intel Atom - Intel Xeon
- Open Source
- BSD-licensed: free and unsupported standalone or commercial solution
DPDK Overview

Runtime environment with low overhead
- Dataplane libraries run in userspace
  1. Memory management
  2. Buffer management
  3. Custom driver
  4. ...
- Environment Abstraction Layer (EAL)

"Easy to use." - Intel
Queue Manager

Fixed-sized ring implemented as table of pointer to any object

Properties:
- FIFO
- Lockless (no active waiting)
- Supports multi consumer/producer enqueue/dequeue scenarios
- Supports bunch-processing of objects
mempool structure:
- Pool of fixed-sized objects
- Uses a ring to store free objects
- Per core cache (optional)
Buffer Manager

mbuf structure used to store network packets

- Created before runtime
- "Allocation": take a free mbuf from a mempool
- "Deallocation": put the mbuf back to the mempool
- Small size to fit in one cache-line (→ mbuf-chaining)

mbuf contains:

1. Metadata: control information, e.g. packet length
2. Pointer to next mbuf
3. Packet data: header and payload
The DPDK creates libraries by creating the EAL:

- Hides environment specifics
- Provides standard programming interface
- Optimized for the available hardware

But does not provide:

- Layer-3 forwarding
- Firewalls
- ...any layer 3 or upper protocol

→ Developer has to port his application to the DPDK
1. Packet Processing using Commodity Hardware

2. Intel’s Dataplane Development Kit

3. Comparison: Intel’s DPDK, netmap, PF_RING DNA
A framework for raw packet I/O, developed by Luigi Rizzo (Università di Pisa)

Feature: works with broad range of soft- and hardware

- Linux and FreeBSD
- Intel 10GbE and 1GbE adapter
- Intel, RealTek, nVidia

Implemented techniques:

- Memory pre-allocation and re-use
- Memory mapping
- Batch processing
- Parallel direct paths (assign CPU core to receiving queue)
PF_RING Direct NIC Access

A framework to capture packets, developed by ntop.

**Feature: zero-copy**

PF_RING DNA maps NIC memory and registers to userland → only one copy operation per packet

**But:** weakness to user misbehaviour (system-crashes)

Implemented techniques:

- Memory pre-allocation and re-use
- Memory mapping (zero-copy)
- Parallel direct paths
### Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel DPDK</th>
<th>netmap</th>
<th>PF_RING</th>
<th>DNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Pre-allocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Memory Mapping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Parallel Direct Paths</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Open Source</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>&quot;Safety&quot;</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Test results show:**
- different frameworks exceed in different use cases [2][4]
- up to 10 times faster than the linux network stack
Sources

Intel DPDK

Intel DPDK
Packet Processing on Intel Architecture.
*Presentation slides, 2012.*

Luigi Rizzo
netmap: a novel framework for fast packet I/O

José Luis García-Dorado et al.
High-Performance Network Traffic Processing Systems Using Commodity Hardware
in: *Data Traffic Monitoring and Analysis, Springer Verlag, 2013.*

www.dpdk.org
Last visited: 06.06.2014
Thank you for your attention! Do you have any questions?