A Look at High-Speed Software Dataplanes and their Upcoming Challenges

Tom Barbette and Alireza Farshin
KTH Royal Institute of Technology
Modular
Software
Dataplanes

Flexible

Reusable bricks
Community

Store

IPSec

Community
Modular Software Dataplanes

- Flexible
- Reusable bricks Community
- Easy development
One Modular Software Dataplane: FastClick

Today's Ecosystem
BESS, VPP, FastClick, ...

Challenges and recent research
Loopback (Simple Forwarding)

```
sudo click --dpdk -- -e
  'FromDPDKDevice(0)
  --> EtherMirror
  --> ToDPDKDevice(0);'
```

- **FromDPDKDevice**: Receives packets from a NIC via DPDK.
- **EtherMirror**: Swaps/Mirrors MAC addresses.
- **ToDPDKDevice**: Sends packets to a NIC via DPDK.
nsrack18 [407] $ sudo click --dpdk -l 0 0 --e "FromDDDkDevice(0)"
  -> MarkIPHeader(14) -> avg :: AverageCounterMP -> EtherMirror
  -> TODDDDkDevice(0); Script(label s, read avg.link_rate, write avg.
  g.reset, wait is, goto s);

avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
  0
avg.link_rate:
variables
CPU=[1-8]

script@dut sudo=true
click --dpdk -l 0-$( $(CPU - 1) ) -- 'FromDPDKDevice(0) -> EtherMirror -> ToDPDKDevice(0);'

import@client fastclick-replay-single-mt trace=/mnt/traces/kth/morning/morning-quad.transformed.pcap

import graph-beautiful
nsrack10 [440] $ npf-run fastclick --cluster client=nsrack17 dut=nsrack18 --test 01-fosdem-fwd.npf --graph-filename 01-fosdem-fwd-results/sv
  g --show-full
Forwarding results

Throughput

FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
Campus trace
Router (A Standard IP Router)
NF chains

FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
Campus trace
2

Today's Ecosystem
Early 2000s...
The Click Modular Router

Eddie Kohler et al.

3200 Citations
What’s the magic?
Single CPU core, router, campus trace

Check the paper for details
Also: NetBricks, NetSlice, DPDK Graph API, ...

Click

BESS
Rebuilt around DPDK

VPP
SSE everywhere

FastClick
Huge legacy

Fork
Which one is best?

Comparing the Performance of State-of-the-Art Software Switches for NFV, Zhang et al., CoNEXT’19

Scatter plots of latency/throughput and of average/standard deviation of latency, under 64B synthetic packets and bidirectional 10Gbps links.
At equivalent features
Simple Forwarding, Single-core at 1200MHz
Challenges for High-Speed Packet Processing + Our Recent Research
Faster link speeds (100/200/400 Gbps)

- Packets are received at a faster pace (every few nanoseconds).
- Accessing memory (DRAM) would kill the performance.
- Inefficient software/hardware would restrict us from processing at high rate.
Per-core performance is not increasing as before

- Demise of Dennard scaling (frequencies are not increasing)
- Less single-thread performance
- More cores
What is Metadata?

- **Packet Metadata**: Information about raw packets/buffer
  - Length
  - Checksum

- **User Metadata or Packet Annotation**: Information produced/used during packet processing
  - Source & Destination IP addresses
  - VLAN ID

Driver

Application
Requires **Two** Copying Operations

**Copy and Conversion**

**DPDK Libraries**

**Copy and Conversion**

**FastClick Model**

1

**Driver Descriptor**

- Driver Metadata

**Poll Mode Driver (PMD)**

**DPDK Descriptor**

- rte_mbuf struct
- Headroom
- Data (Buffer)

**Copied Descriptor**

- Application Metadata
- Buffer Address

**Overlayed Descriptor**

- DPDK Metadata
- Application Metadata
- Buffer Address

**Application**
Requires **Two** Copying Operations

**Copy and Conversion**

- **Copy and Conversion**
- **DPDK Libraries**
  - DPDK Descriptor
    - rte_mbuf struct
    - Headroom
    - Data (Buffer)

**Point and Cast**

- **Overlayed Descriptor**
  - DPDK Metadata
  - Application Metadata
  - Buffer Address

Requires **One** Copy Operation but Carries Unnecessary Fields

**FastClick Model**

1

**BESS Model**

2

Requires **Two** Copying Operations

**Copy and Conversion**

- **Copy and Conversion**
- **Driver Descriptor**
  - Driver Metadata

Poll Mode Driver (PMD)

**Copy and Conversion**

- **Copy and Conversion**
- **Application**
  - Copied Descriptor
    - Application Metadata
    - Buffer Address
  - Overlaid Descriptor
    - DPDK Metadata
    - Application Metadata
    - Buffer Address

**Requirements**

- **Copy and Conversion**
  - Requires **One** Copy Operation

**Copy and Conversion**

- **Copy and Conversion**
  - Requires **Two** Copying Operations

**Point and Cast**

- **Point and Cast**
  - Requires **One** Copy Operation

**FastClick Model**

1

**BESS Model**

2
X-Change

- Exchanging buffers with DPDK
- Provides custom buffers to DPDK drivers
- Prevents any extra operation
- Fewer in-flight buffers
- Avoid allocating/releasing mbufs
- Implemented via conversion functions (requires linking)

tbarbette/xchange
Metadata Management Models

Simple Forwarding
Throughput

FastClick
Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz
Fixed-size Packets
Mellanox ConnectX-5 (MLX5 Driver *)
*Without vectorized PMD
PacketMill

A tool that uses the available information to build a customized- and optimized-binary for the input NF

- X-Change (using customized DPDK buffers)
- Source-code modifications (embedding constants+graph and devirtualizing)
- IR-code modifications (reordering data structures)

Better Metadata Management
Reduce the Cost of Flexibility
PacketMill

Simple Forwarding Throughput

FastClick
Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz
Fixed-size Packets
Mellanox ConnectX-5 (MLX5 Driver *)
*Without vectorized PMD

Check out our extended abstract and upcoming paper at ASPLOS'21:
PacketMill: Toward per-core 100-Gbps Networking

aliireza/packetmill
Don’t write your network dataplane from scratch, use a modular software dataplane!

Better, use FastClick+PacketMill!

- [tbarbette/fastclick](https://github.com/tbarbette/fastclick)
- [aliireza/packetmill](https://github.com/aliireza/packetmill)
Don’t write your network dataplane from scratch, use a modular software dataplane!

Better, use FastClick+PacketMill!

- tbarbette/fastclick
- aliireza/packetmill
Using Conversion Functions rather than Direct Assignment

- **DPDK Implementation (MLX5)**
  
  ```c
  pkt->vlan_tci = rte_be_to_cpu_16(cqe->vlan_info);
  ```

- **X-Change Implementation (MLX5)**
  
  ```c
  xchg_set_vlan_tci(pkt, rte_be_to_cpu_16(cqe->vlan_info));
  ```

- **Conversion Functions**
  
  ```c
  /* Default DPDK */
  void xchg_set_vlan_tci(struct xchg* pkt, uint16_t vlan_tci) {
    ((struct rte_mbuf*)pkt)->vlan_tci = vlan_tci;
  }

  /* Custom Implementation */
  void xchg_set_vlan_tci(struct xchg* pkt, uint16_t vlan_tci) {
    SET_VLAN_ANNO((Packet*)pkt, vlan_tci);
  }
  ```
Conclusion

FastClick comes with lots of great features

Provides good performance

Well-integrated with NPF, which enables easy prototyping

Multi-hundred-Gbps networking means staying in L1 and L2

Deep-optimize your pipeline with PacketMill!

X-Change allows to avoid the rte_mbuf, and directly spawn your descriptor

- tbarbette/fastclick
- aliireza/packetmill
Metadata Management Models

1. FastClick (Copying)
   - Copies the user metadata data from rte_mbuf

2. BESS, FastClick (Overlaying)
   - Overlays the user metadata data with rte_mbuf

1 + 2. VPP (Copying+Overlaying)
   - Overlays the user metadata data with rte_mbuf
   - Copies some of the fields
## Metadata Management Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FastClick</td>
<td>Copies the user metadata data from rte_mbuf</td>
</tr>
<tr>
<td>2 BESS, FastClick</td>
<td>Overlays the user metadata data with rte_mbuf</td>
</tr>
<tr>
<td>1 + 2 VPP</td>
<td>Overlays the user metadata data with rte_mbuf; Copies some of the fields</td>
</tr>
<tr>
<td>PacketMill</td>
<td>Provides custom buffers to DPDK drivers; Prevents any extra operation</td>
</tr>
</tbody>
</table>
How to Make the Most out of the Current Hardware?

- Better load balancing
- Avoid unnecessary memory accesses
- Optimize software
Example of improvements

Thread traversal analysis
Example of improvements

Userlevel clock

- No timestamp: 15.26 GBps
- User: 14.05 GBps
- vDSO: 12.47 GBps
What does FastClick have on top of the others?

- Thread vector
- Userlevel timing

But it lacks:

- Metadata Liveness Analysis (BESS)
- SSE Instructions* (VPP)

* Their real impact with many scattered different flows should to be proven.
Forwarding results

Latency

FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
Campus trace
Router (single core)

FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
PacketMill

1. X-Change API
   - Customizing Metadata
   - Click Source + xchg.o
   - Linking
   - DPDK Source

2. Source-Code Modifications
   - Config File
   - NF Configuration
   - PacketMill

3. IR-Code Modifications
   - Optimized IR Code
   - Compile LTO
   - Specialized Binary
   - Merged IR Code
   - Reordering Data Structures

Configuration-based Optimizations

Compile

NF Configuration

Optimized Click Source

Click Binary

xchg.o +
Software Dataplanes

Flexible

Cheap

Outsourcing
nsrack10 [406] % sudo click --dpdk -- e "FromDPDKDevice(0)
  -> MarkIPHeader(14) -> ICMPFingerResponder -> EtherMirror
  -> ToDPDKDevice(0);"

nsrack17 [1132] % < ~/workspace/fosdem (master+4)[18:30:07]
47

nsrack17 [1132] $ sudo click --dpdk --e "FromDump(trace.pcap) -> Pad
-> ToDPDKDevice(0)"

EAL: Detected 16 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/rtc/mp_socket
EAL: Selected I/O mode "PA"
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: Probe PCI driver: mlx5_pci (15b3:1017) device: 0000:11:00.0 (socket 0)
common_ml5: RTE_MEM is selected.
mlx5_pci: Size 0xFFFF is not power of 2, will be aligned to 0x10000.
EAL: Probe PCI driver: mlx5_pci (15b3:1017) device: 0000:11:00.1 (socket 0)
mlx5_pci: Size 0xFFFF is not power of 2, will be aligned to 0x10000.
EAL: No legacy callbacks, legacy socket not created
Initialzizing DPDK
expensive Packet::put: have 0 wanted 225
expensive Packet::put: have 0 wanted 1396
expensive Packet::put: have 0 wanted 1386
expensive Packet::put: have 0 wanted 1306
expensive Packet::put: have 0 wanted 1386
expensive Packet::put: have 0 wanted 1396
nsrack18 [407] $ sudo click --dpdk -l 0 -o --e 'FromDPDKDevice(0) []
-> MarkTPHeader(14) -> avg : AverageCounterMP -> EtherMirror
-> ToDPDKDevice(0); Script(label s, read avg.link_rate, write av
g.reset, wait is, goto s);

47

nsrack17" 10:33 06-Jan-21