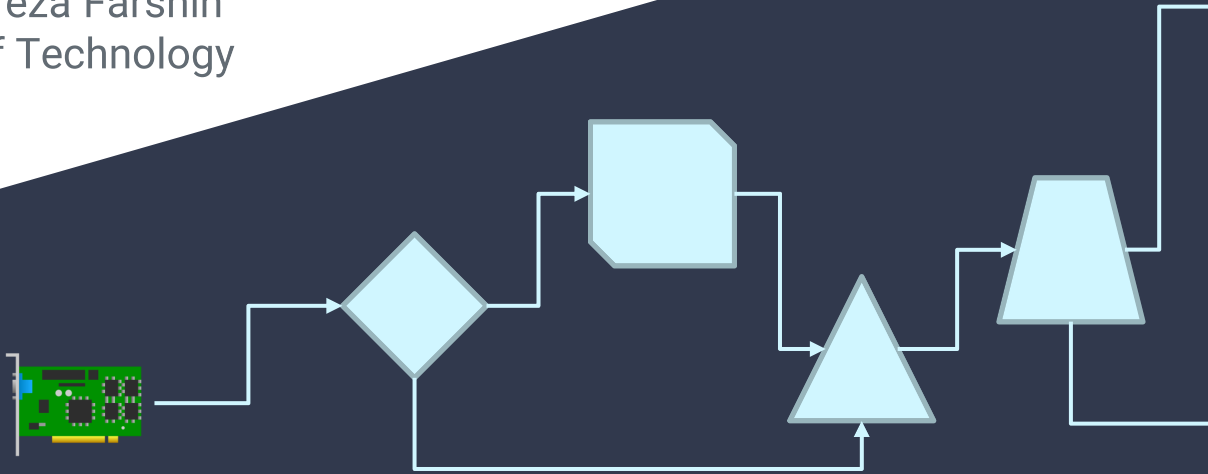
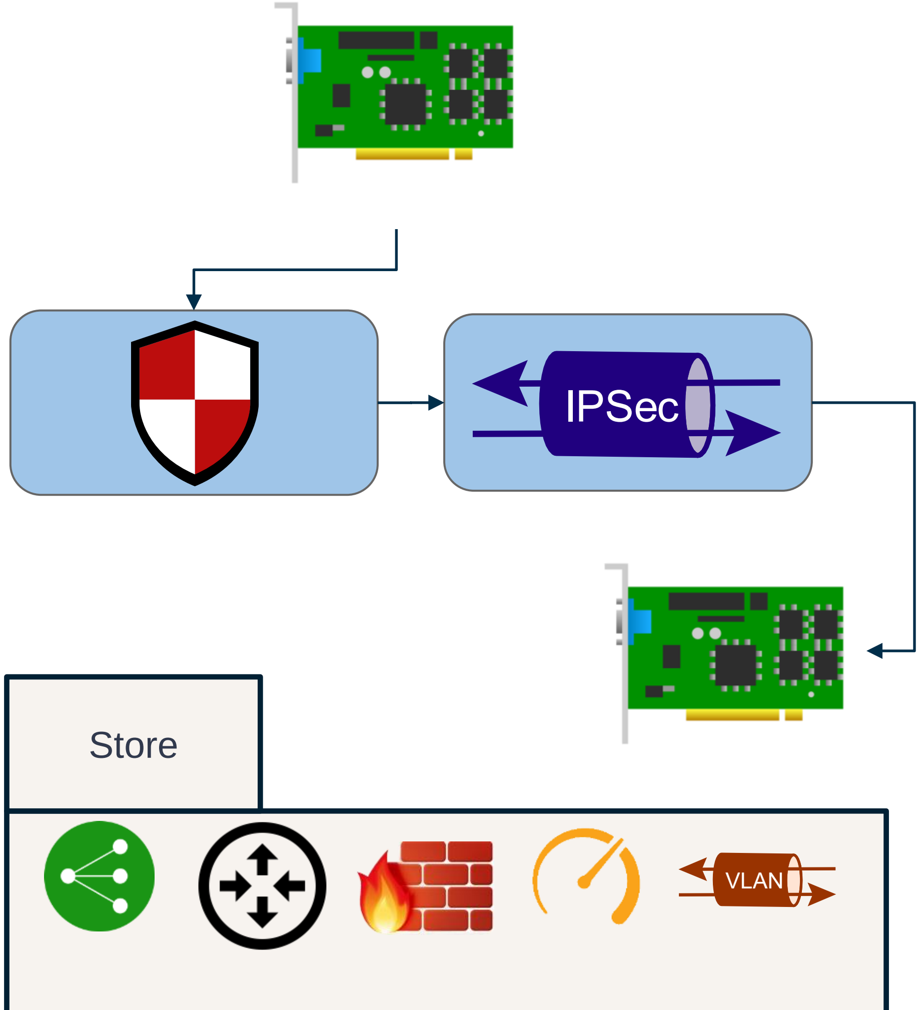
A Look at High-Speed Software Dataplanes and their Upcoming Challenges

Tom Barbette and Alireza Farshin
KTH Royal Institute of Technology



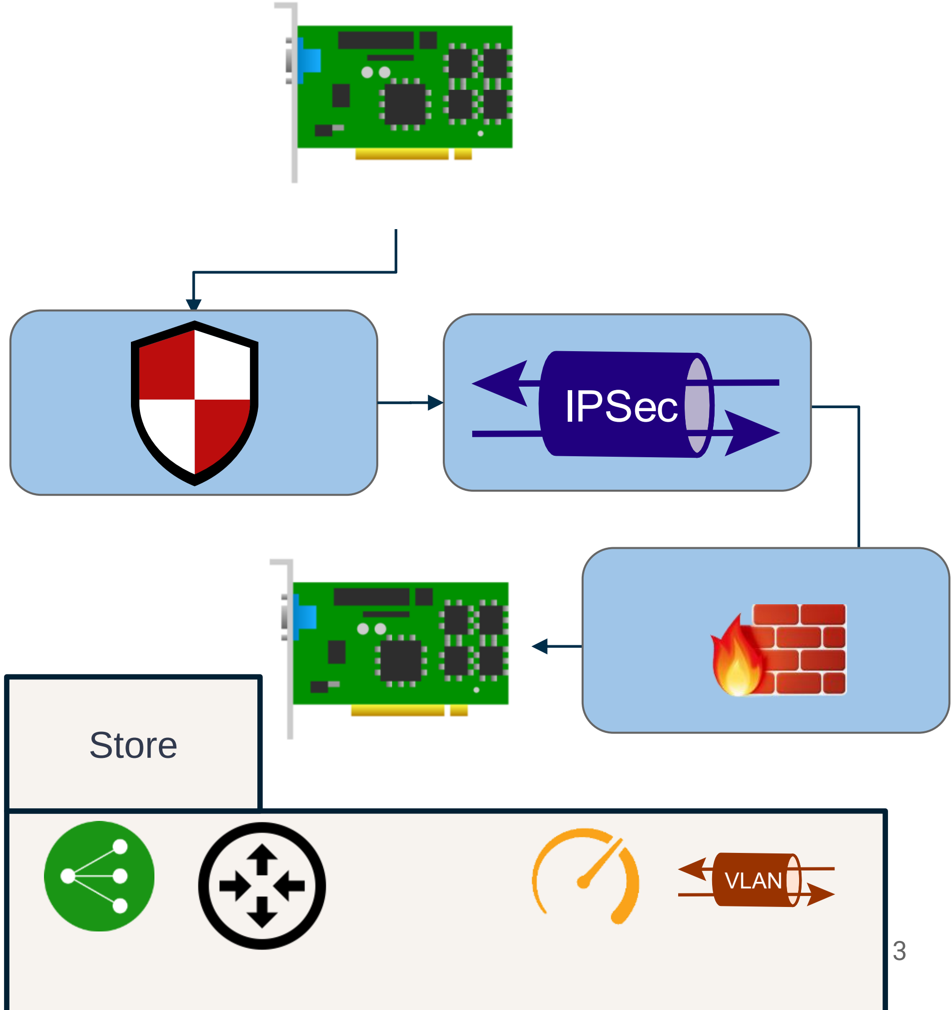
Modular Software Dataplanes

- Flexible
- Reusable bricks
  Community



Modular Software Dataplanes

- Flexible
- Reusable bricks
  Community
- Easy development



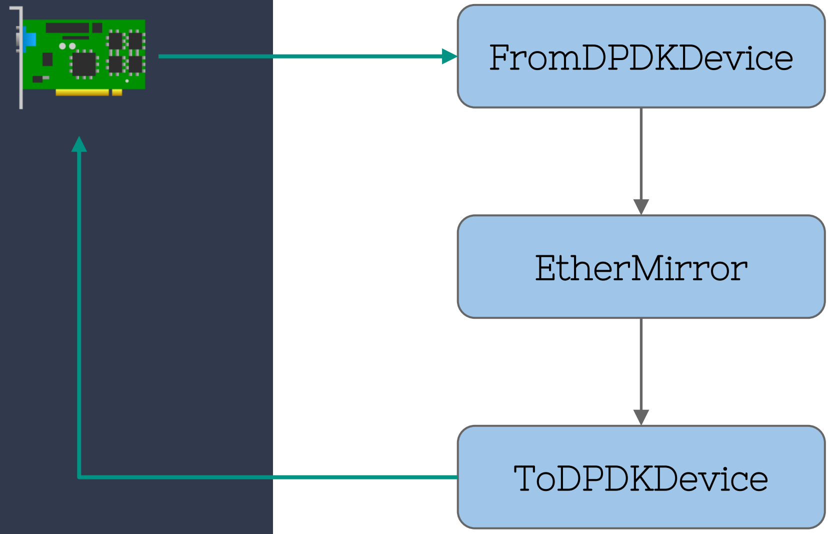
1 One Modular Software Dataplane: FastClick

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

3 Challenges and recent research

Loopback (Simple Forwarding)

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



Receives packets from a NIC via DPDK

Swaps/Mirrors MAC addresses

Sends packets to a NIC via DPDK

```
nslrack18 [367] % █                                        < ~/workspace/fastclick (master↑1|)[17:51:47]

[0] 0:ssh*                                                      "nslrack17" 17:51 06-Jan-21
```

```
0
nslrack18 [407] % sudo click --dpdk -l 0-0 -- -e 'FromDPDKDevice(0)
        -> MarkIPHeader(14) -> avg :: AverageCounterMP -> EtherMirror
        -> ToDPDKDevice(0); Script(label s, read avg.link_rate, write av
g.reset, wait 1s, goto s);'

 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
```

```
nslrack17 [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/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: Probe PCI driver: mlx5_pci (15b3:1017) device: 0000:11:00.0 (socke
t 0)
common_mlx5: 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 (socke
t 0)
mlx5_pci: Size 0xFFFF is not power of 2, will be aligned to 0x10000.
EAL: No legacy callbacks, legacy socket not created
Initializing DPDK
expensive Packet::put; have 0 wanted 225
expensive Packet::put; have 0 wanted 1386
expensive Packet::put; have 0 wanted 1386
expensive Packet::put; have 0 wanted 1306
expensive Packet::put; have 0 wanted 1386
^C%
nslrack17 [1133] % sudo click --dpdk -- -e "FromDump(trace.pcap) -> Ens
ureDPDKBuffer -> Pad -> ReplayUnqueue(LIMIT 1000000, QUICK_CLONE 1) ->
ToDPDKDevice(0); DPDKInfo(1048575);";
```

[0] 0:zsh* "nslrack17" 18:34 06-Jan-21

```
%variables
CPU=[1-8]

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

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

%import graph-beautiful
```

"01-fosdem-fwd.npf" 9L, 267C 5,19 All

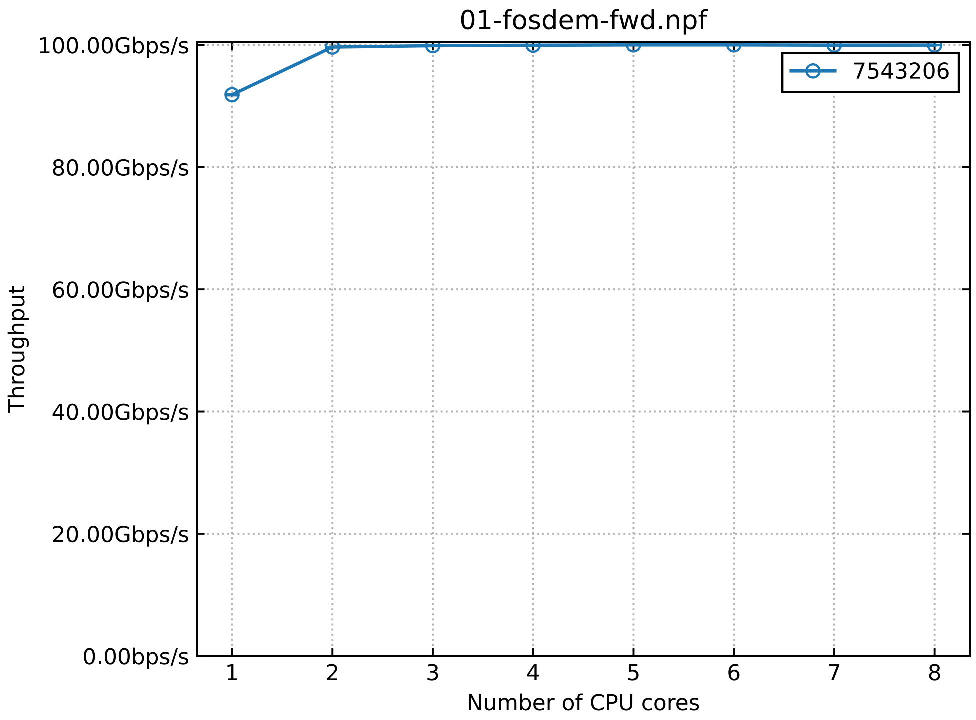
[0] 0:ssh* "nslrack17" 18:35 06-Jan-21

```
nslrack18 [440] % npf-run fastclick --cluster client=nslrack17 dut=nslrack18 --test 01-fosdem-fwd.npf --graph-filename 01-fosdem-fwd-results/.sv
g --show-full
```

```
[0] 0:ssh*Z                                                                                   "nslrack17" 19:06 06-Jan-21
```

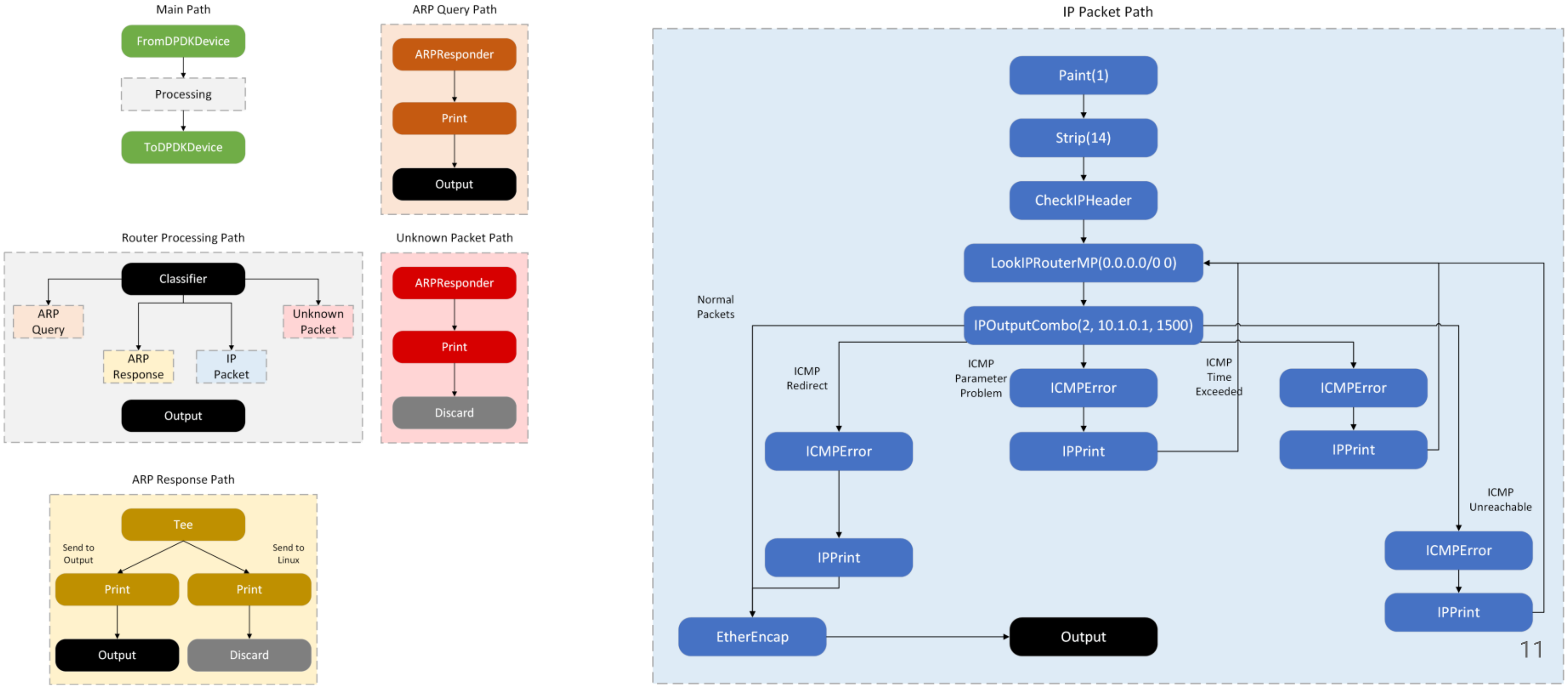
Forwarding results

Throughput

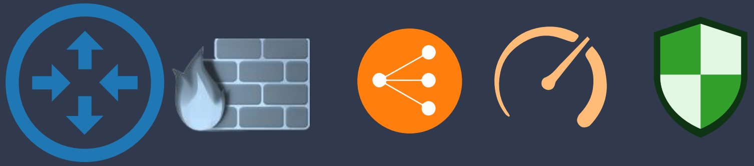


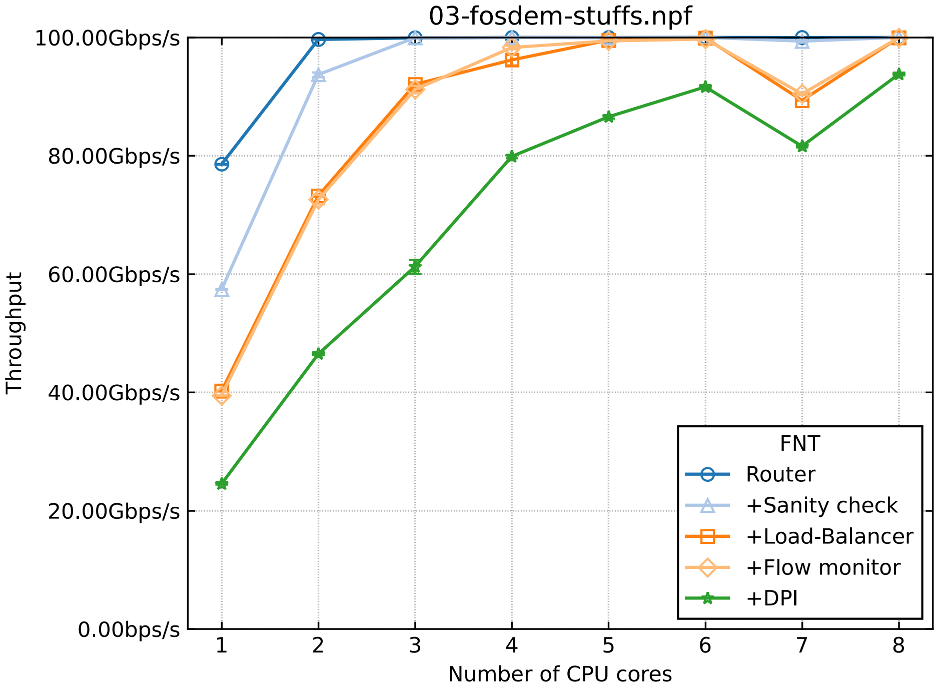
FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
Campus trace

Router (A Standard IP Router)



NF chains





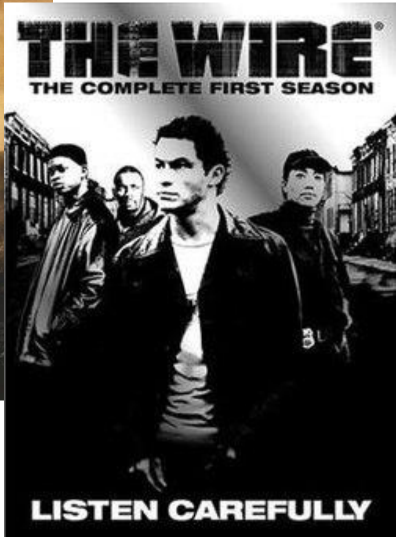
FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
Campus trace

2

Today's Ecosystem

Early 2000s...

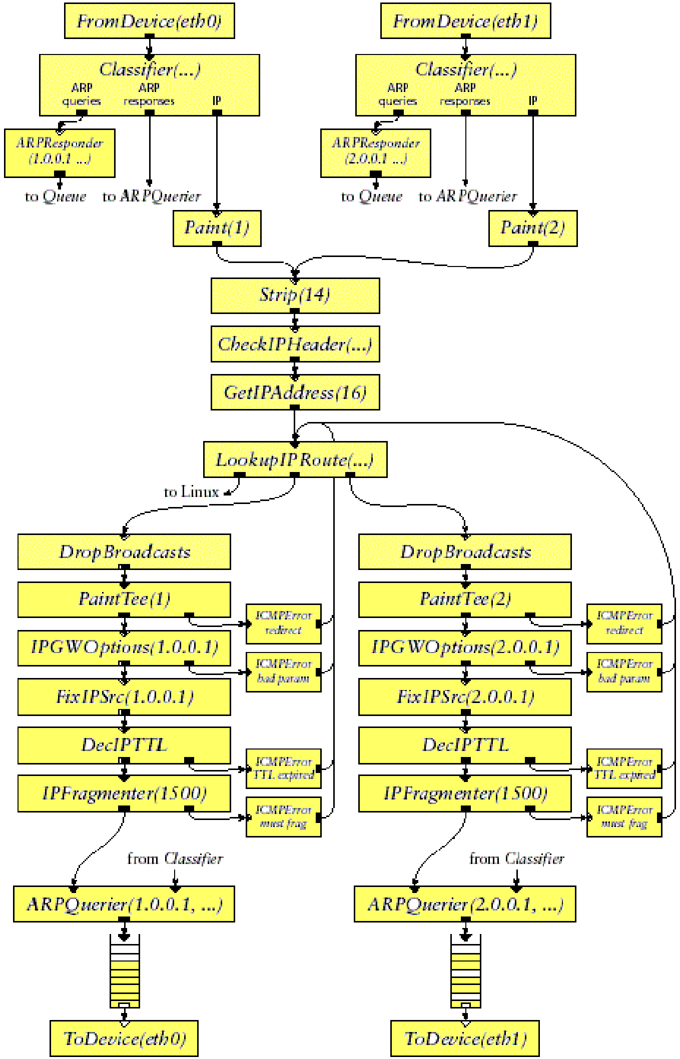

THE LORD OF THE RINGS
THE TWO TOWERS
THE JOURNEY CONTINUES


THE WIRE
THE COMPLETE FIRST SEASON
LISTEN CAREFULLY

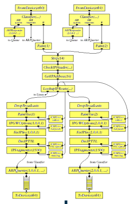
The Click Modular Router

Eddie Kohler *et al.*

3200 Citations



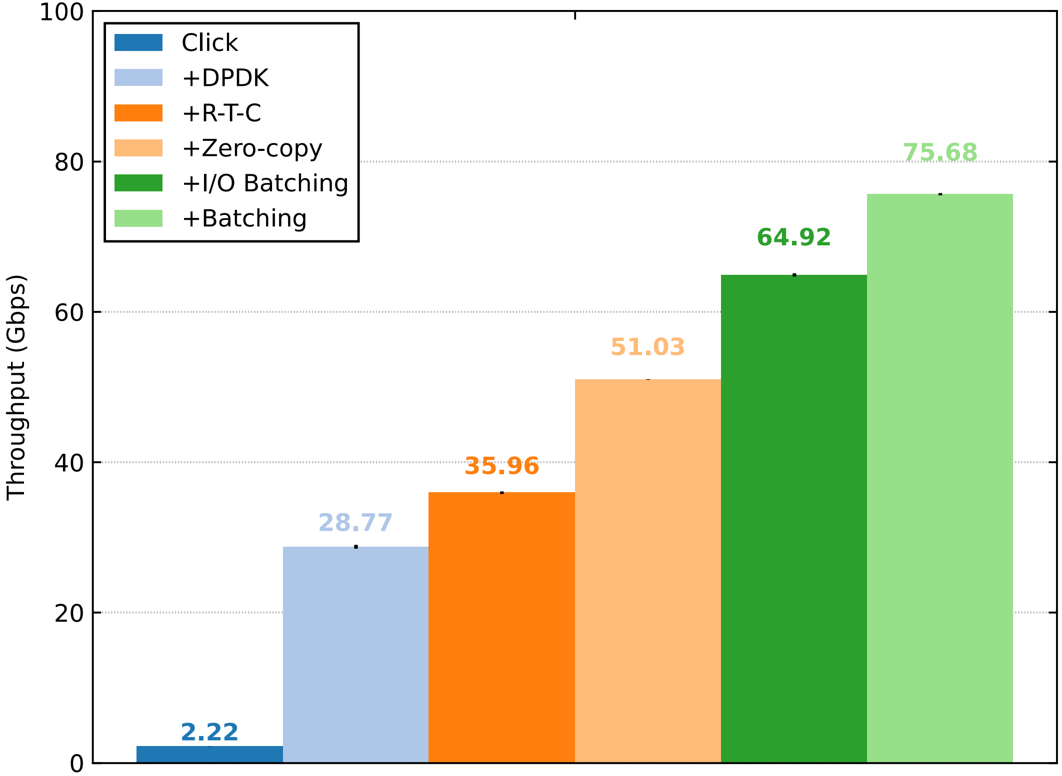
Click



Fork

FastClick

What's the magic?
Single CPU core, router, campus trace

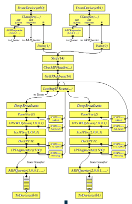


tbarbette/fastclick

Check the paper for details

https://bit.ly/3bzcfjG

Click



Fork

BESS

Rebuilt around DPDK

VPP

SSE everywhere

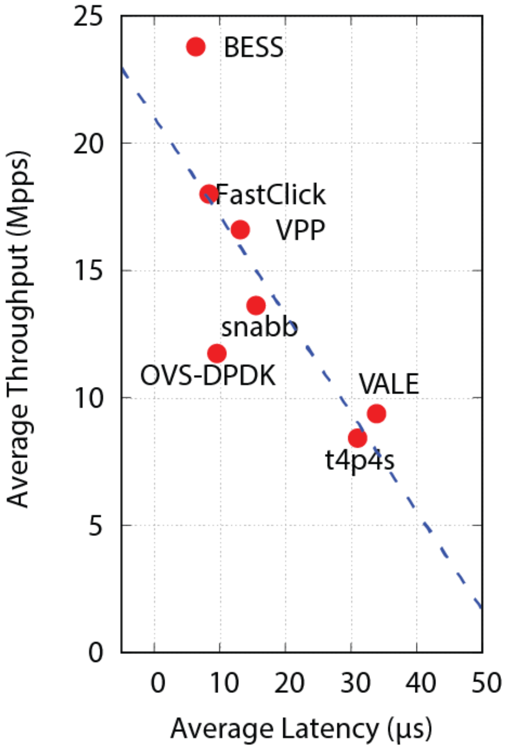
FastClick

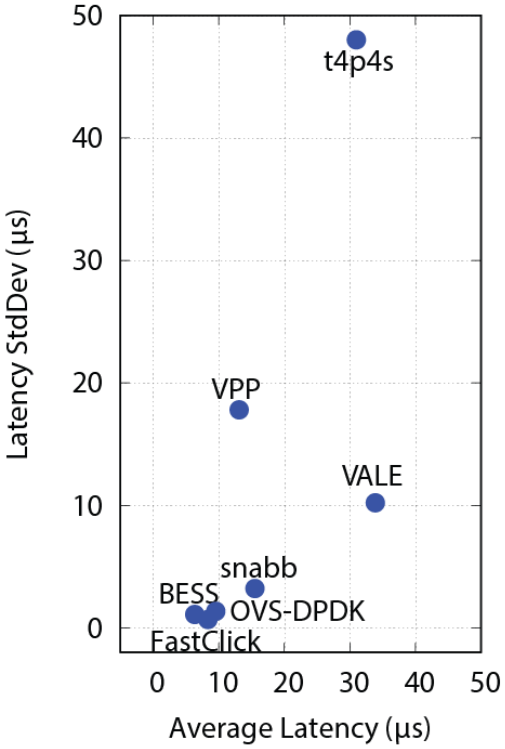
Huge legacy

Also : NetBricks, NetSlice, DPDK Graph API, ...

Which one is best?

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

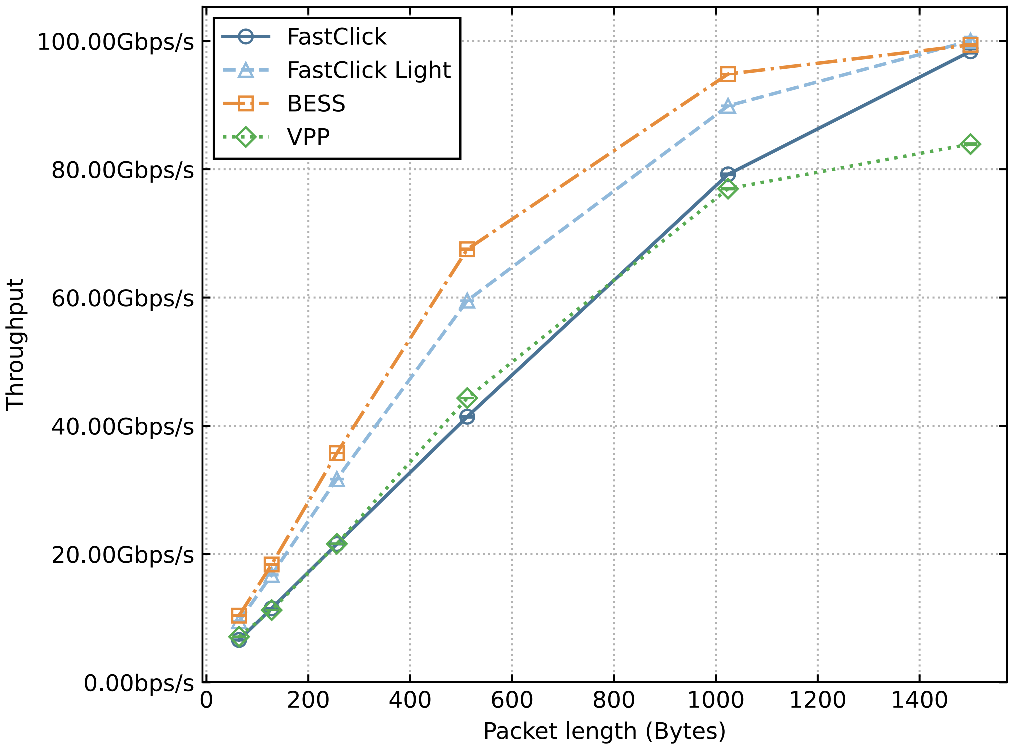




Scatter plots of latency/throughput and of average/standard deviation of latency, under 64B synthetic packets and bidirectional 10Gbps links.

At equivalent features

Simple Forwarding, Single-core at 1200MHz

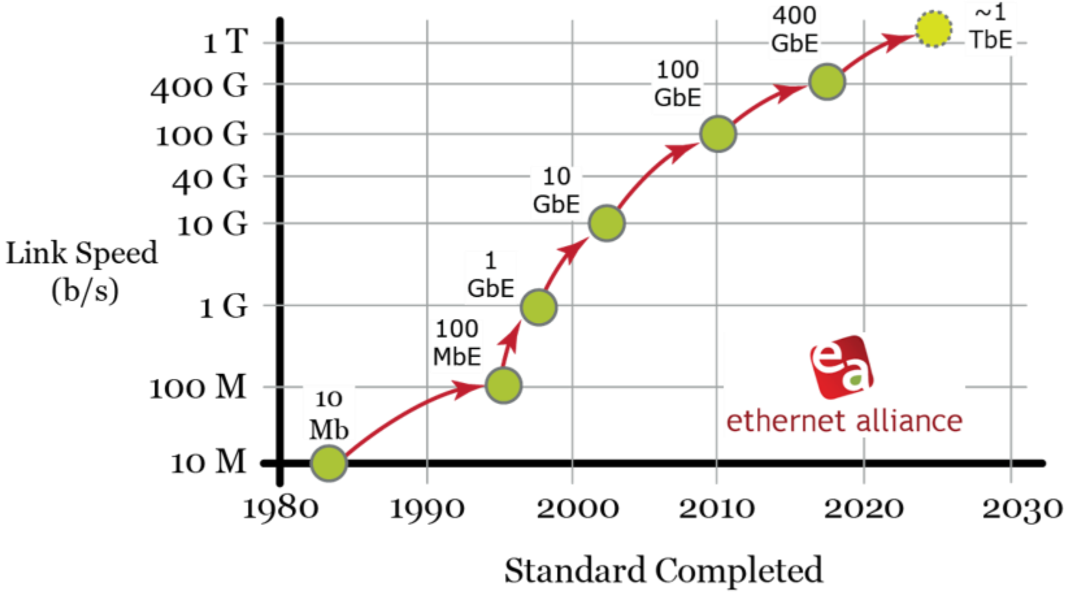


3

Challenges for High-Speed Packet Processing
+
Our Recent Research

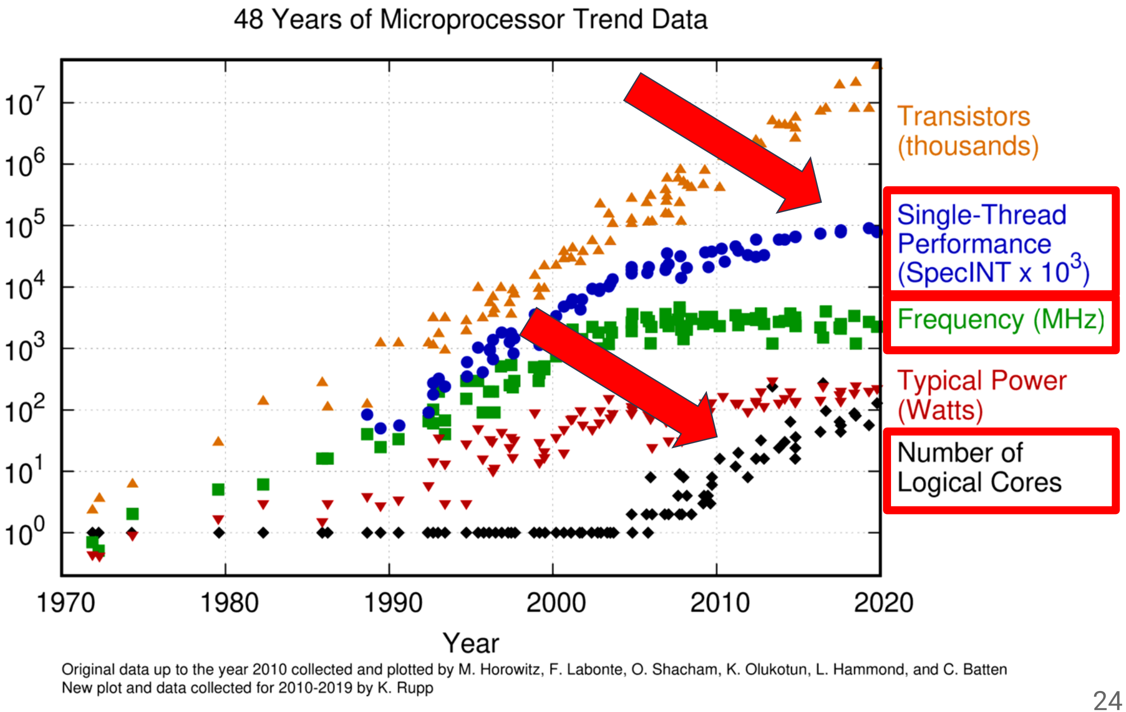
Faster link speeds (100/200/400 Gbps)

- 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

Driver

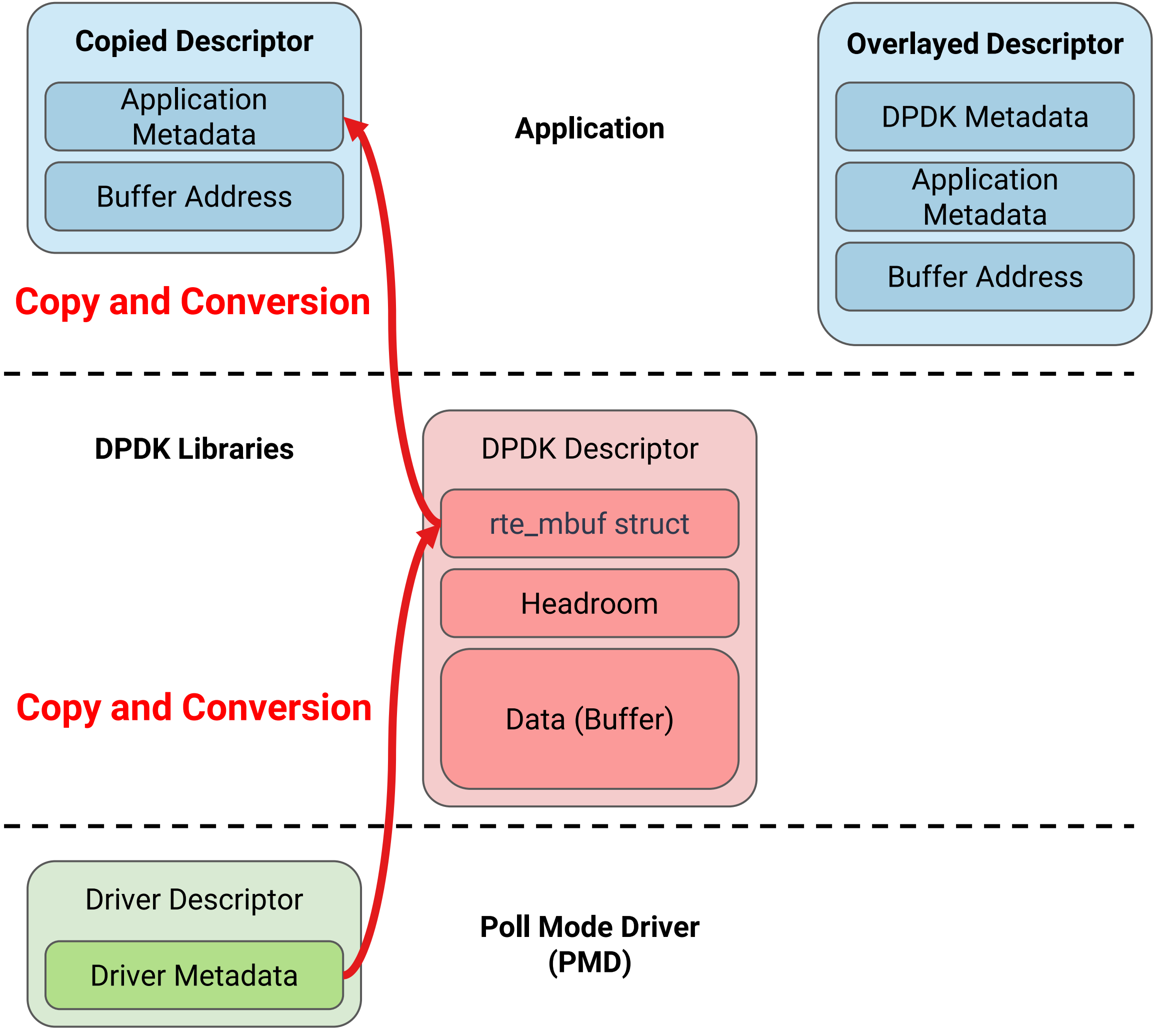
- **User Metadata or Packet Annotation:** Information produced/used during packet processing
  - Source & Destination IP addresses
  - VLAN ID

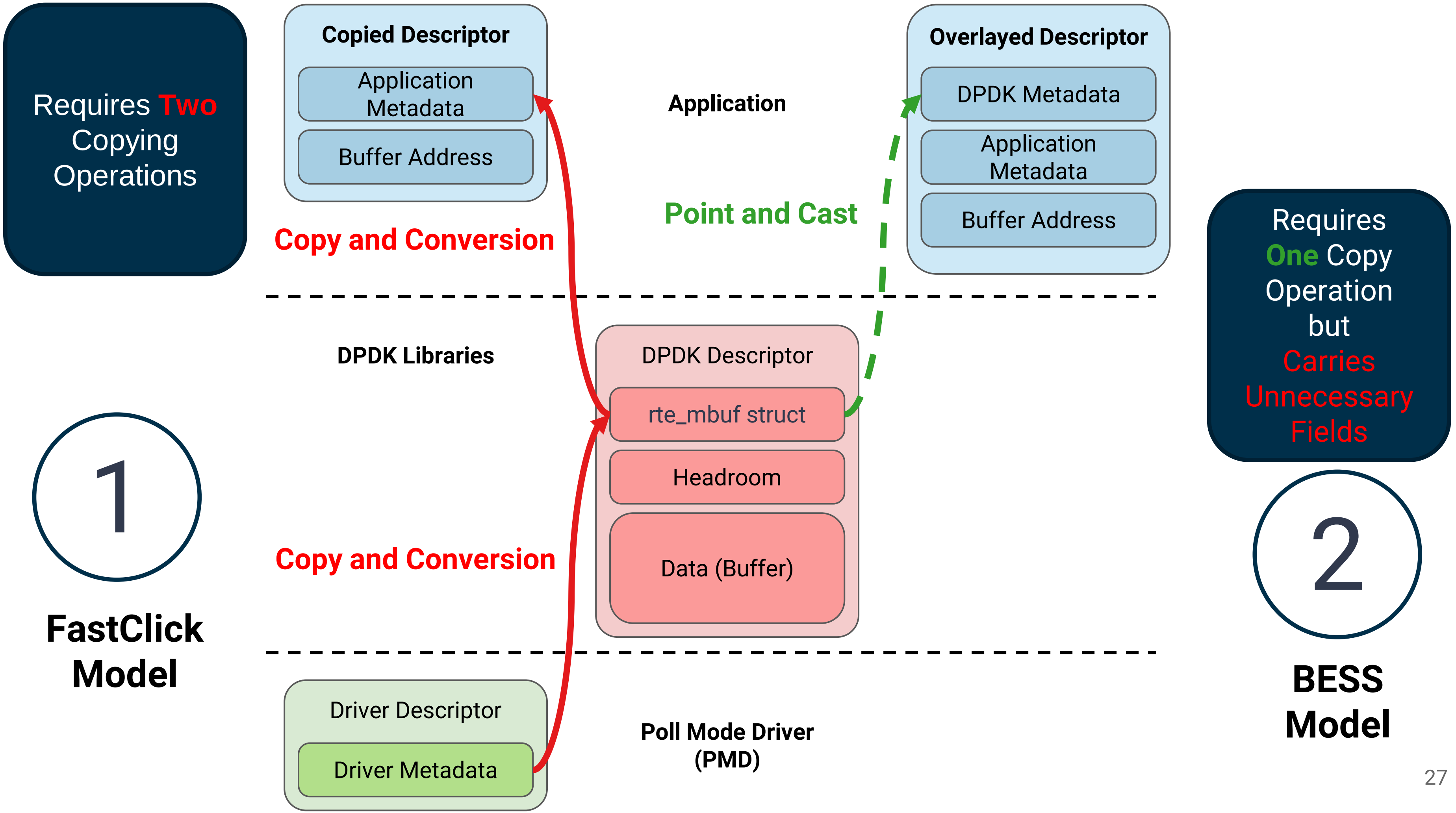
Application

Requires Two Copying Operations

1

FastClick Model

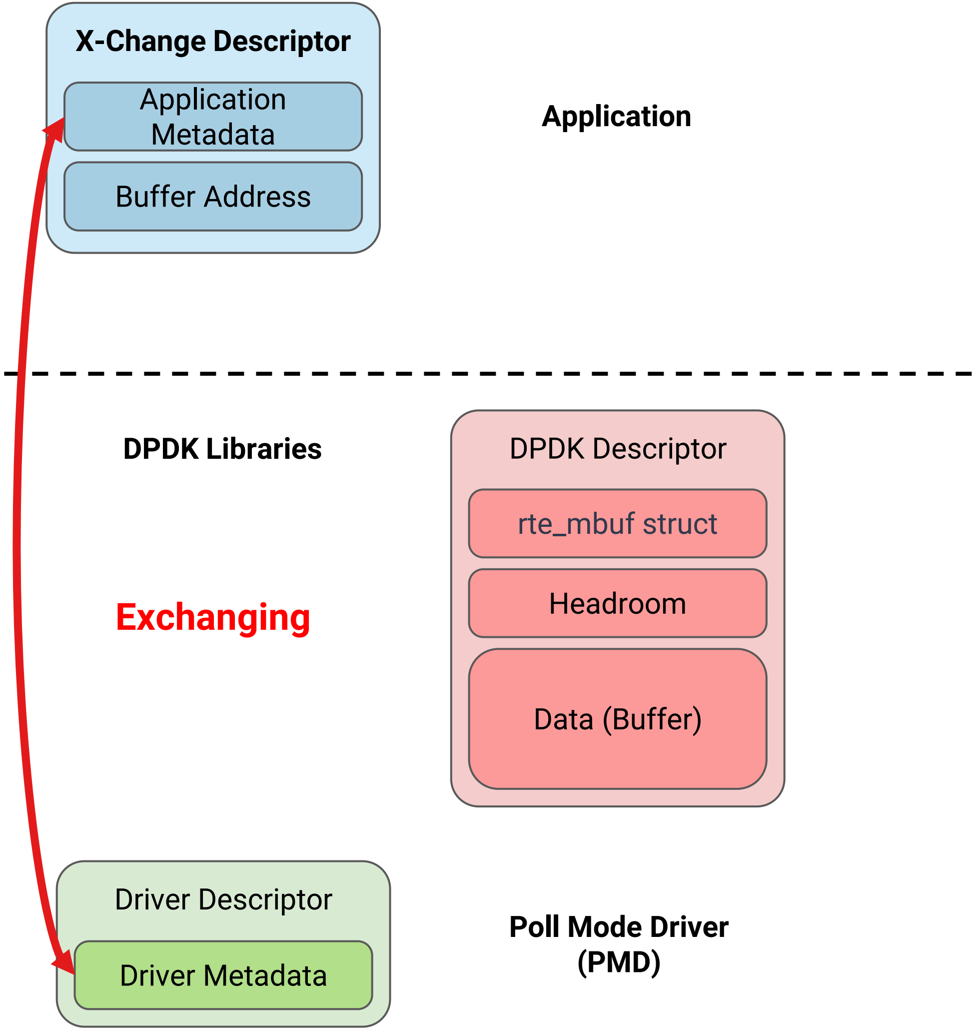



Requires Two Copying Operations
Copied Descriptor
Application Metadata
Buffer Address
Application
Overlayed Descriptor
DPDK Metadata
Application Metadata
Buffer Address
Point and Cast
Copy and Conversion
Requires One Copy Operation but Carries Unnecessary Fields
DPDK Libraries
DPDK Descriptor
rte_mbuf struct
Headroom
Data (Buffer)
1
Copy and Conversion
2
FastClick Model
BESS Model
Driver Descriptor
Driver Metadata
Poll Mode Driver (PMD)

X-Change

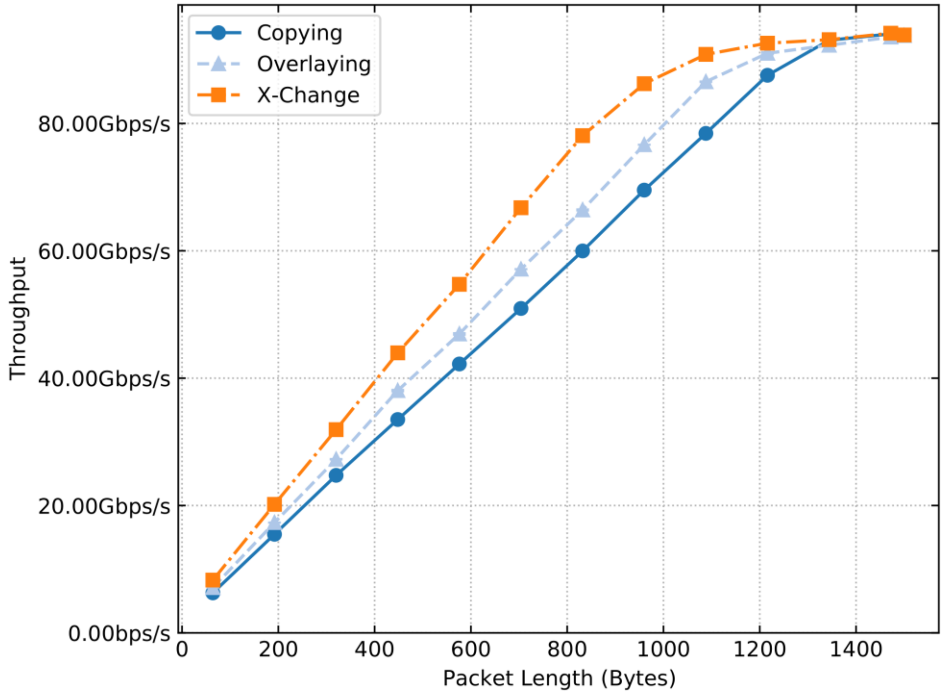
- Exchanging buffers with DPDK
- Provides custom buffers to DPDK drivers
- Prevents any extra operation
- Fewer in-flight buffers
- Avoid allocating/releasing mbufs
- Implemented via conversion functions (requires linking)

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

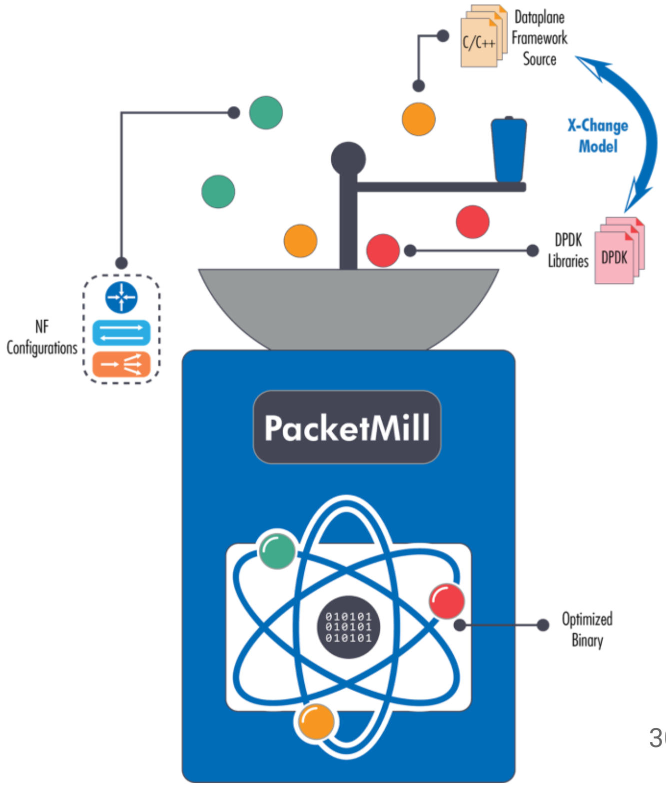
A tool that uses the available information to build a customized- and optimized-binary for the input NF

- X-Change (using customized DPDK buffers)

Better Metadata Management

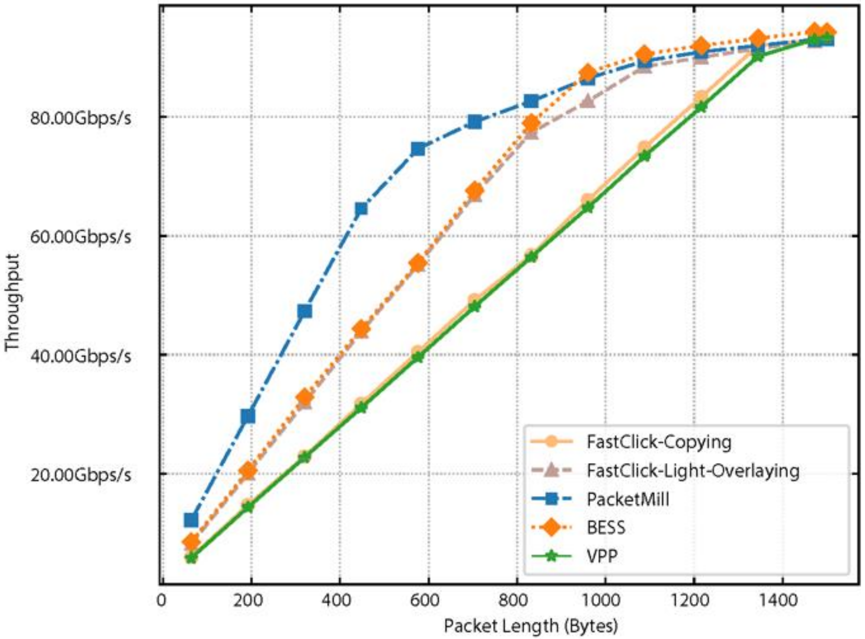
- Source-code modifications (embedding constants+graph and devirtualizing)
- IR-code modifications (reordering data structures)

Reduce the Cost of Flexibility



PacketMill

Simple Forwarding
Throughput

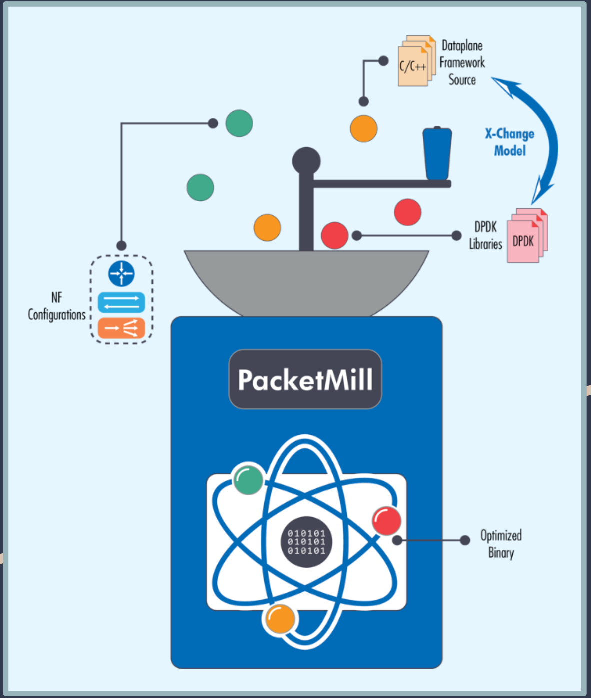


Check out our extended abstract and upcoming paper at ASPLOS'21:
[PacketMill: Toward per-core 100-Gbps Networking]

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

[aliireza/packetmill]

Conclusion



Don't write your network dataplane from scratch, use a modular software dataplane!

Better, use **FastClick+PacketMill!**

tbarbette/fastclick

aliireza/packetmill

Q&A

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

Better, use **FastClick+PacketMill!**

tbarbette/fastclick

aliireza/packetmill

X-Change Implementation

Using Conversion Functions rather than Direct Assignment

- DPDK Implementation (MLX5)

```
pkt->vlan_tci = rte_be_to_cpu_16(cqe->vlan_info);
```

- X-Change Implementation (MLX5)

```
xchg_set_vlan_tci(pkt, rte_be_to_cpu_16(cqe->vlan_info));
```

- Conversion Functions

```
/* 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	2	1 + 2
FastClick (Copying)	BESS, FastClick (Overlaying)	VPP (Copying+Overlaying)
• Copies the user medata data from rte_mbuf	• Overlays the user medata data with rte_mbuf	• Overlays the user medata data with rte_mbuf • Copies some of the fields

Metadata Management Models

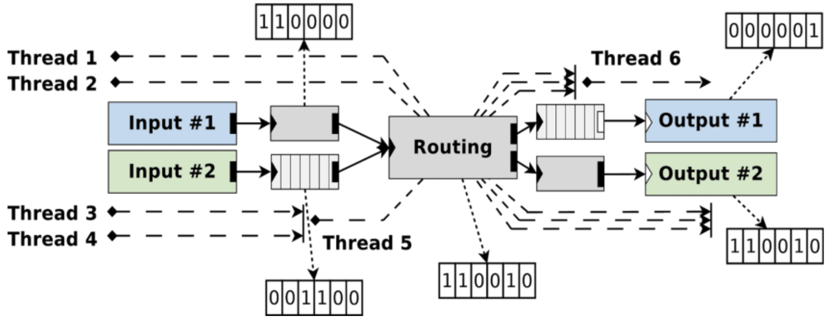
①	②	① + ②	
FastClick (Copying)	BESS, FastClick (Overlaying)	VPP (Copying+Overlaying)	PacketMill (X-Change)
• Copies the user medata data from rte_mbuf	• Overlays the user medata data with rte_mbuf	• Overlays the user medata data with rte_mbuf • Copies some of the fields	• Provides custom buffers to DPDK drivers • Prevents any extra operation

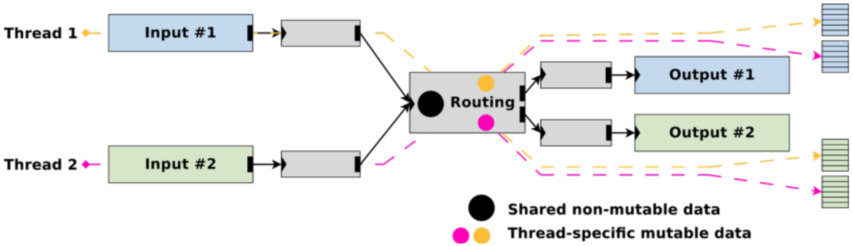
How to Make the Most out of the Current Hardware?

- Better load balancing
- Avoid unnecessary memory accesses
- Optimize software

Example of improvements

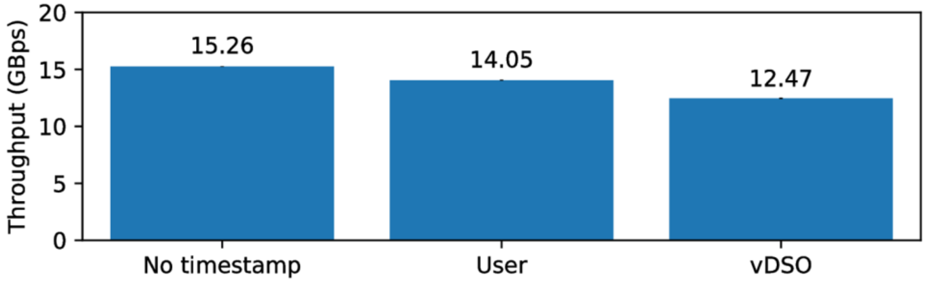
Thread traversal analysis





Example of improvements

Userlevel clock



What does FastClick have on top of the others?

- Thread vector
- Userlevel timing

But it lacks:

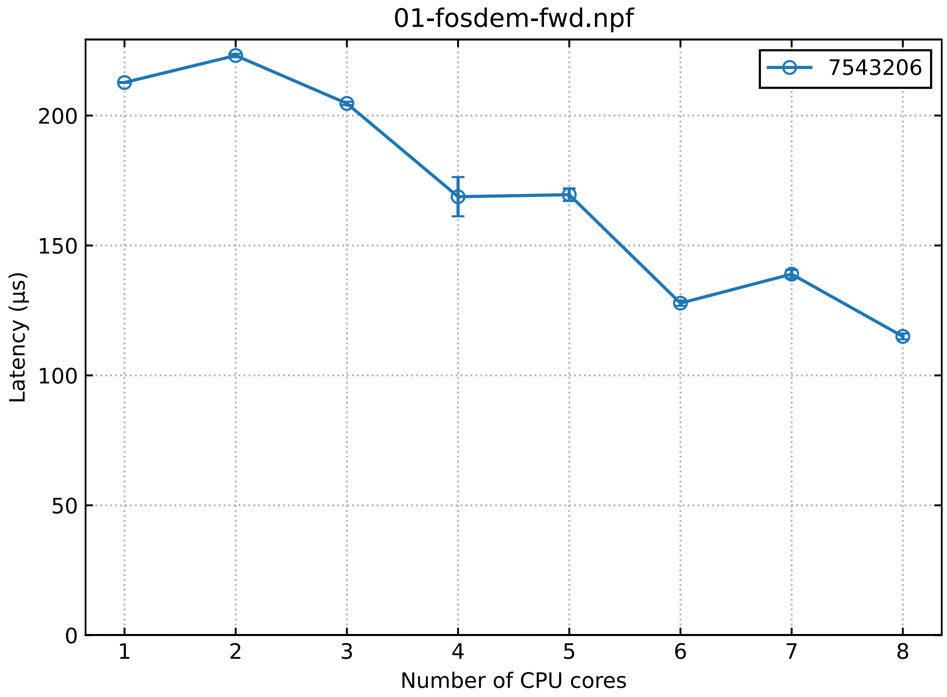
- Metadata Liveness Analysis (BESS)
- SSE Instructions* (VPP)

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

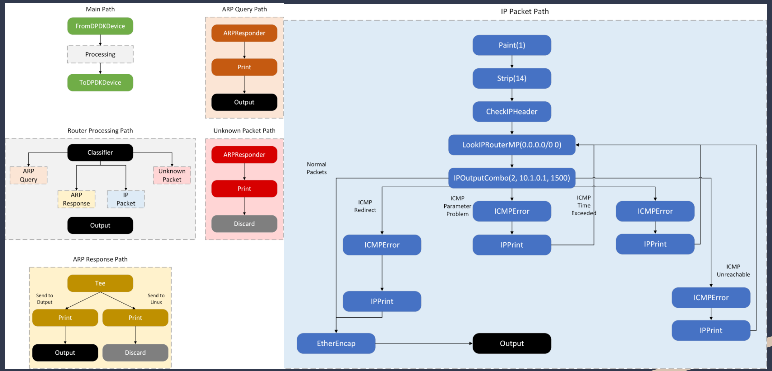
Forwarding results

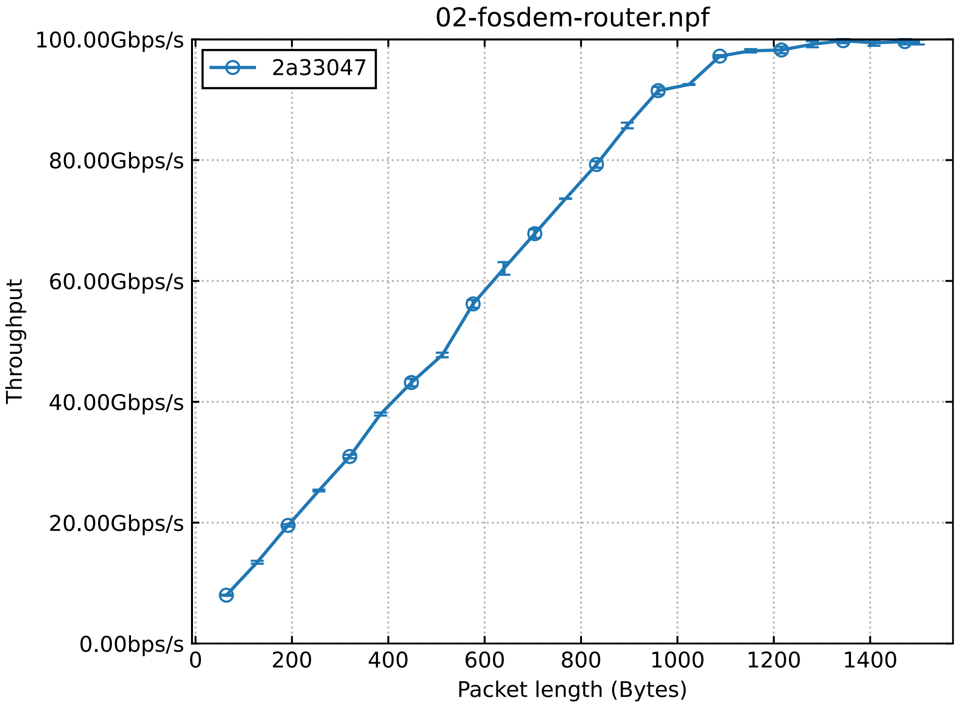
Latency

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



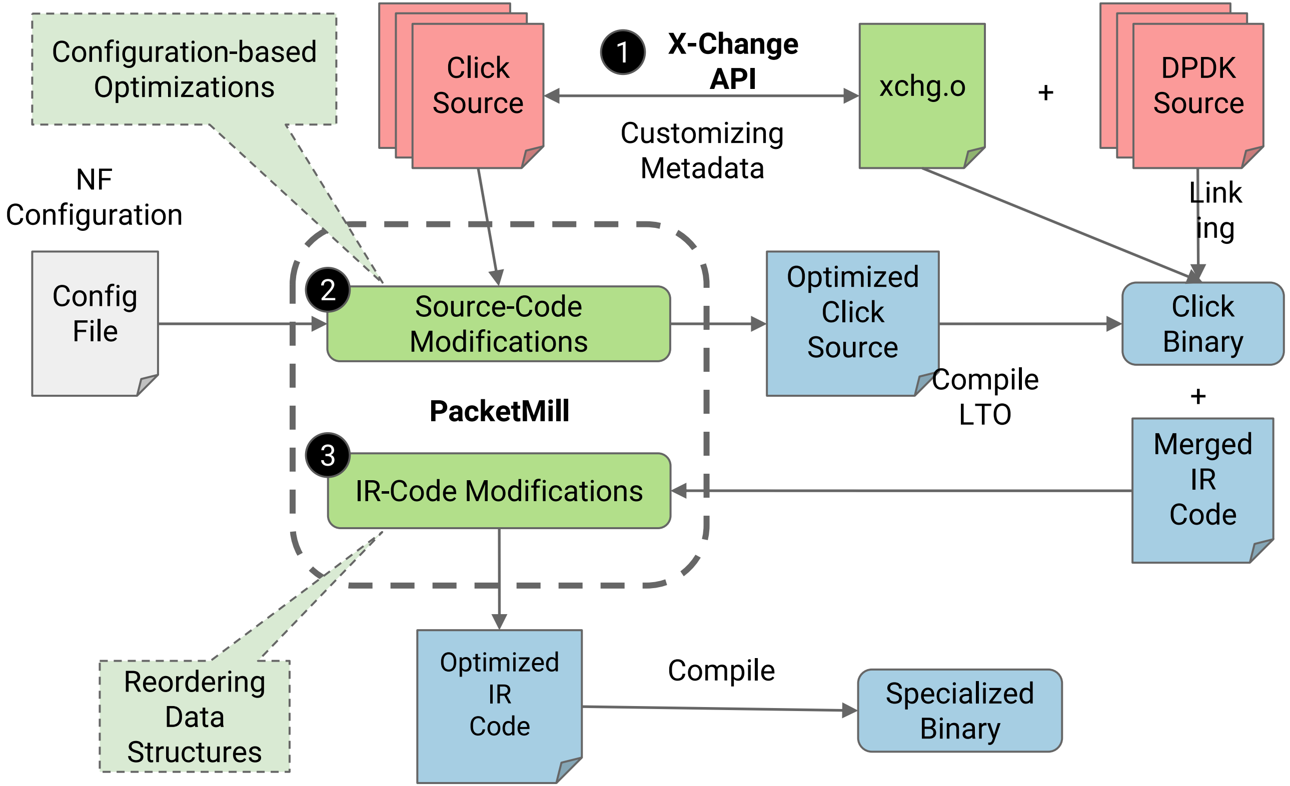
Router (single core)





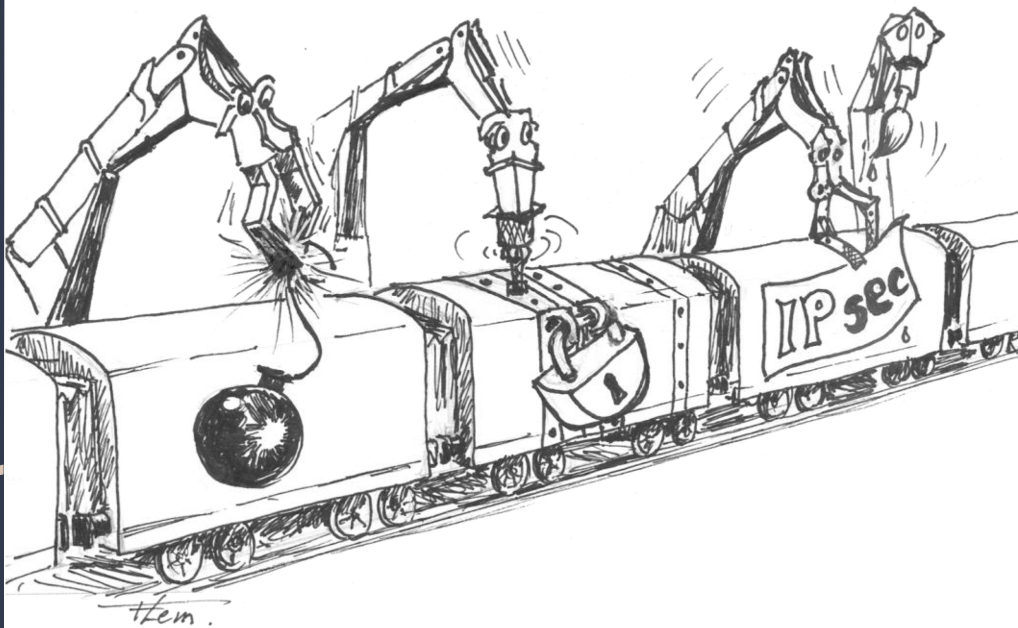
FastClick
Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz

PacketMill


Configuration-based Optimizations
Click Source
1
X-Change API
Customizing Metadata
xchg.o
+
DPDK Source
NF Configuration
Link ing
Config File
2
Source-Code Modifications
Optimized Click Source
Click Binary
Compile LTO
PacketMill
+
3
IR-Code Modifications
Merged IR Code
Reordering Data Structures
Optimized IR Code
Compile
Specialized Binary

Software Dataplanes

- Flexible
- Cheap
- Outsourcing



```
nslrack18 [406] % sudo click --dpdk -- -e 'FromDPDKDevice(0)
        -> MarkIPHeader(14) -> ICMPPingResponder -> EtherMirror
        -> ToDPDKDevice(0);'
```

```
nslrack17 [1132] %          < ~/workspace/fosdem (master|+4)[18:30:07]
```

```
[0] 0:ssh*                                      "nslrack17" 18:32 06-Jan-21
```

```
51(1406,1472,1458) ack 1791135481 win 693
0.000000: 133.11.15.121.52227 > 13.33.16.163.443: R 411349358:411349358(
0,60,40) ack 4001056433 win 2047
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705078224:1705079672(
1448,1514,1500) ack 2349629 win 501
0.000000: 131.153.230.78.443 > 130.98.83.18.56712: . 3439665451:34396668
57(1406,1472,1458) ack 1791135481 win 693
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705079672:1705081120(
1448,1514,1500) ack 2349681 win 501
0.000000: 131.153.230.78.443 > 130.98.83.18.56712: . 3439666857:34396682
63(1406,1472,1458) ack 1791135481 win 693
0.000000: 134.103.1.72.60497 > 185.174.116.179.8801: . 429237141:4292371
41(0,66,52) ack 3049607033 win 4093
0.000000: 132.200.161.156.54210 > 192.16.125.174.22: . 2749682336:274968
3784(1448,1514,1500) ack 646764945 win 1444
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705081120:1705082568(
1448,1514,1500) ack 2349681 win 501
0.000000: 131.153.230.78.443 > 130.98.83.18.56712: . 3439668263:34396696
69(1406,1472,1458) ack 1791135481 win 693
0.000000: 132.100.158.216.443 > 130.98.68.145.53387: . 303522128:3035221
28(0,66,52) ack 1111696526 win 972
0.000000: 131.153.230.78.443 > 130.98.83.18.56712: . 3439669669:34396710
75(1406,1472,1458) ack 1791135481 win 693
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705082568:1705084016(
1448,1514,1500) ack 2349681 win 501
0.000000: 134.79.56.221.56294 > 13.107.4.50.80: . 4185404275:4185404275(
0,60,40) ack 670067871 win 10057
0.000000: 134.103.1.72.60497 > 185.174.116.179.8801: . 429237141:4292371
41(0,66,52) ack 3049607391 win 4082
0.000000: 134.103.1.72.60497 > 185.174.116.179.8801: . 429237141:4292371
41(0,66,52) ack 3049608094 win 4060
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705084016:1705085464(
1448,1514,1500) ack 2349681 win 501
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705085464:1705086912(
1448,1514,1500) ack 2349681 win 501
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705086912:1705088360(
1448,1514,1500) ack 2349681 win 501
0.000000: 134.57.35.16.34176 > 89.44.250.42.22: . 1705088360:1705089808(
1448,1514,1500) ack 2349681 win 501
nslrack18 [407] % sudo click --dpdk -l 0-0 -- -e 'FromDPDKDevice(0)
        -> MarkIPHeader(14) -> avg :: AverageCounterMP -> EtherMirror
        -> ToDPDKDevice(0); Script(label s, read avg.link_rate, write av
g.reset, wait 1s, goto s);'
```

```
nslrack17 [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/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: Probe PCI driver: mlx5_pci (15b3:1017) device: 0000:11:00.0 (socke
t 0)
common_mlx5: 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 (socke
t 0)
mlx5_pci: Size 0xFFFF is not power of 2, will be aligned to 0x10000.
EAL: No legacy callbacks, legacy socket not created
Initializing DPDK
expensive Packet::put; have 0 wanted 225
expensive Packet::put; have 0 wanted 1386
expensive Packet::put; have 0 wanted 1386
expensive Packet::put; have 0 wanted 1306
expensive Packet::put; have 0 wanted 1386
```



```
[0] 0:ssh*                                      "nslrack17" 18:33 06-Jan-21
```