DPDK
DATA PLANE DEVELOPMENT KIT

THE LINUX FOUNDATION

DPDK SUMMIT CHINA 2017

主办方： intel

参与方： 腾讯云 ZTE 美团云 Panabit 太一星晨 Balance Your Networks UnitedStack 云杉网络 Yunshan Networks

协办方： SDNLAB 专注网络创新技术 视频支持方： IT大咖说

Embedded Network Architecture Optimization Based on DPDK

Lin Hao
T1 Networks

Agenda

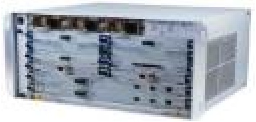
- **Our History —** What is an embedded network device
- **Challenge to us —** Requirements for device today
- **Our solution —** T1 unique embedded network architecture（T1-System）
  - Model of “embedded network architecture”
  - History of T1-system
  - Business layer of T1-system
  - An optimization case —— dual-socket system
  - T1-system as a NFV

Our History

T1 Networks —

"Professional application delivery & High-performance fusion of network security products"

Harbor Networks Corp.



Product: Router

HW: Freescale + Intel NP

SW: vxworks + uCode

Venustech Corp.

Product: UTM

HW: Cavium OCTEON

SW: cvm excutiveSDK

T1 networks Corp.

Product: ADC

HW: X86

SW: Linux+Netmap

Product: NGFW

HW: X86

SW: Linux+**DPDK**

2000 — 2006 — 2013 — 2015

Challenge to our system

Situation

1. Falling cost on network bandwidth

10Gbps 100Gbps
40Gbps 10/100/1000 Mbps

2.Hardware is varied and iteration fast

Xeon Atom I350 82599
Core XL710 X552
RRC

3.Features expansion

VPN Policy Anti-virus QOS IPS Compress

require for our system

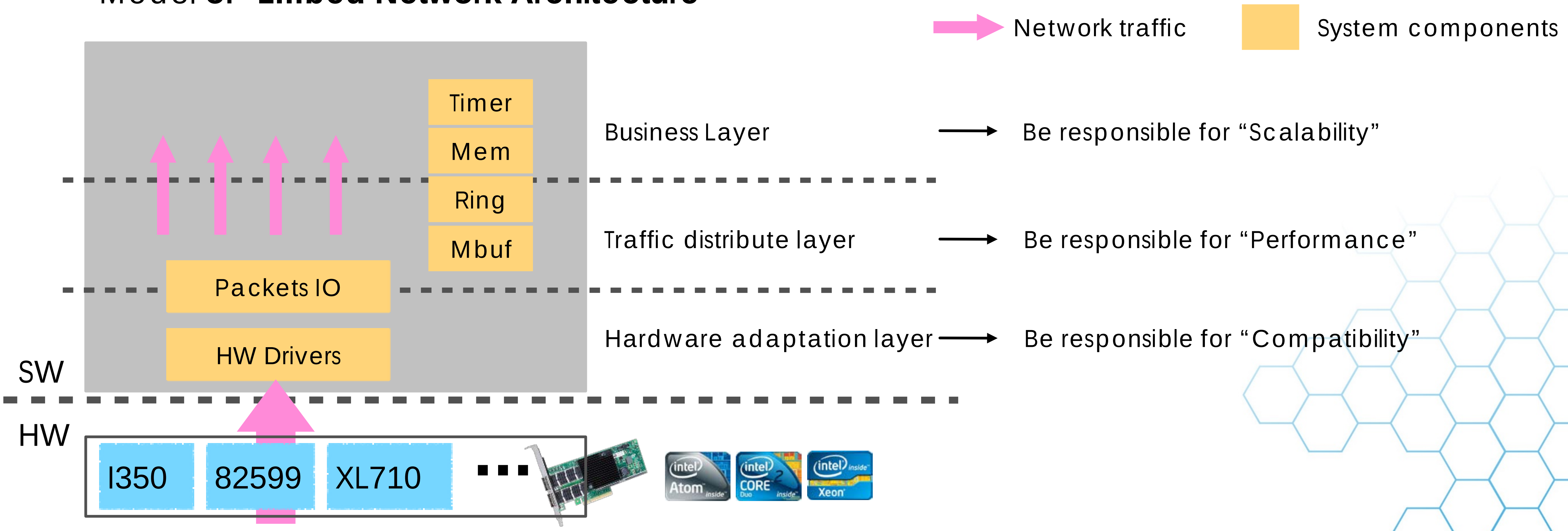
Performance !

Compatibility !

Scalability !

Model of ENA

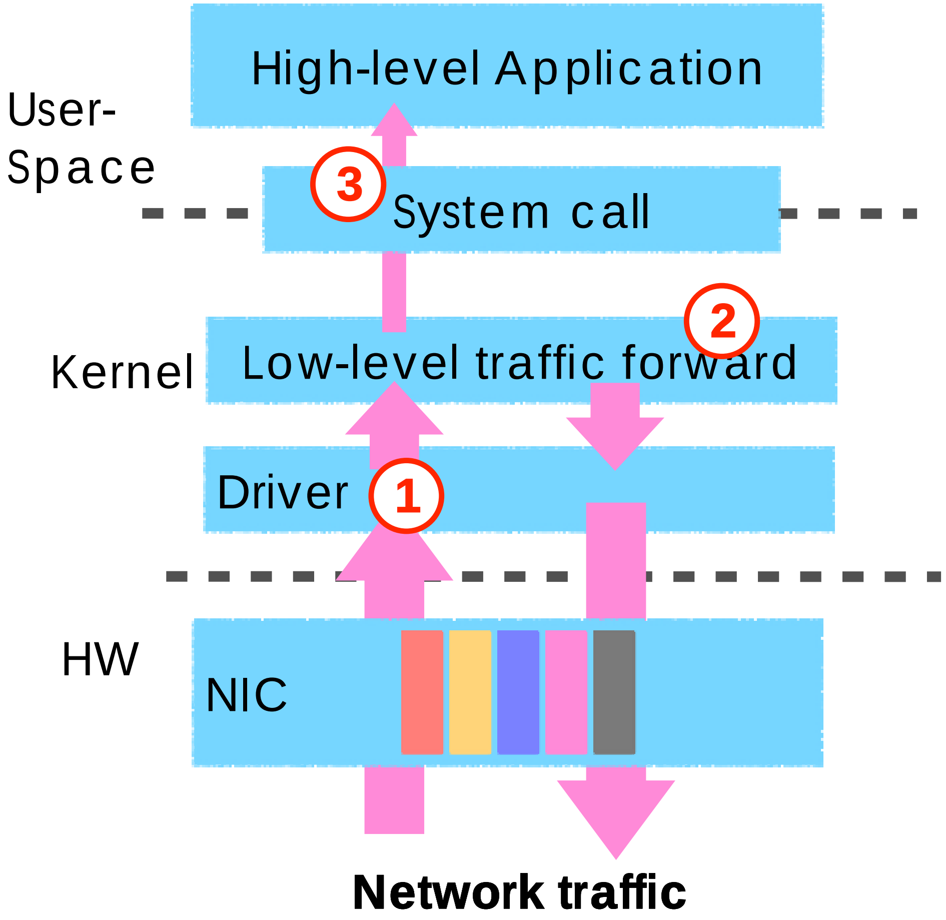
Model **of “Embed Network Architecture”**



History of “T1-system”

- 1st Generation —— “kernel driver based” system
- 2nd Generation —— Muti-Core MIPS64
- 3rd Generation —— “Dispatcher-application” system
- 4th Generation —— “Balanced-dispatcher” DPDK-equipped system
- 5th Generation —— “DPDK+FPGA” system
- Why we need DPDK？How to use DPDK？

1st Gen—Kernel driver based



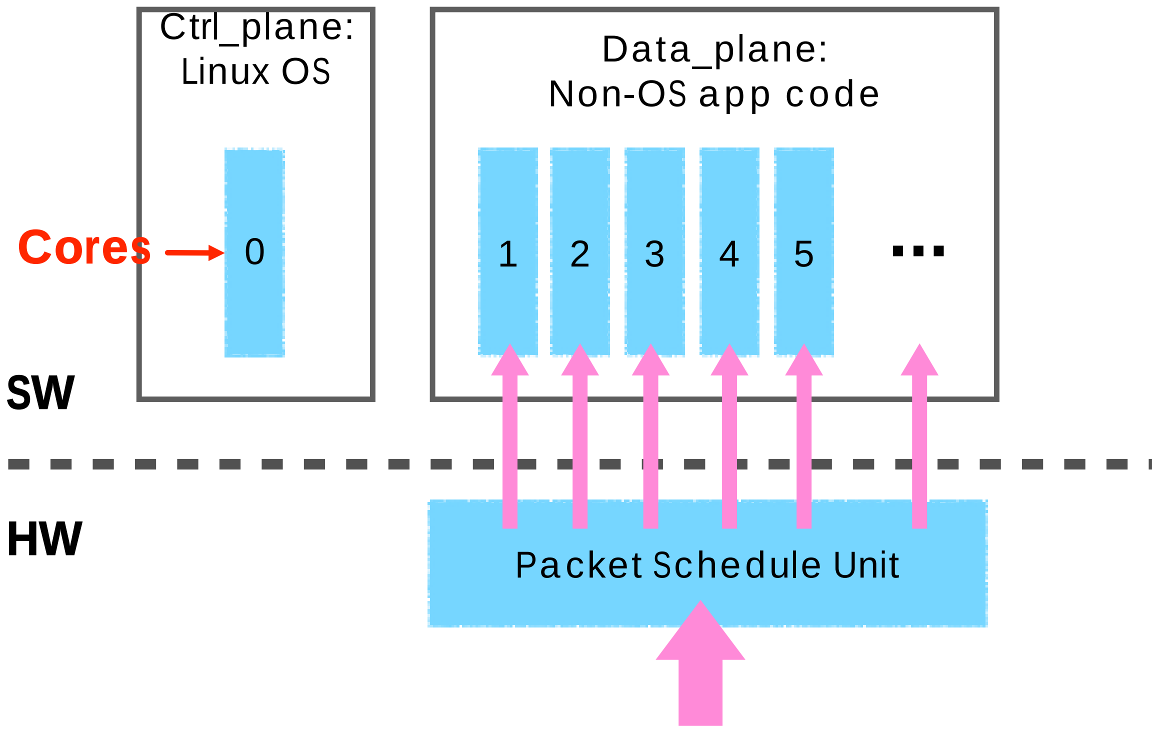
Advantage :

Easy to get……

Problem:

1. Bottleneck of Linux IRQ
2. Difficult to develop and optimize
3. Inefficient system call

2rd Gen—Muti-Core MIPS64



Advantage：

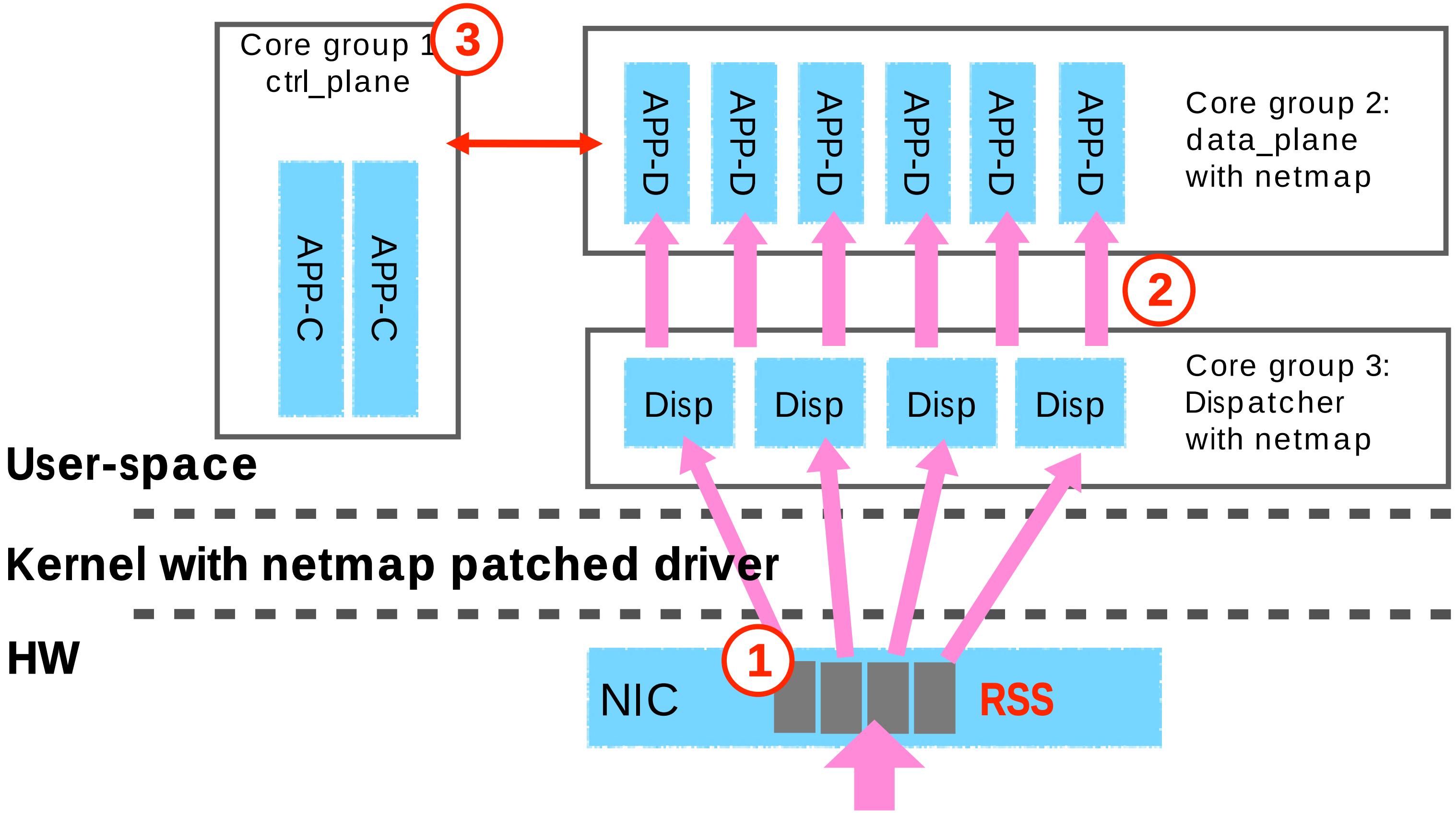
Excellent throughput performance

Problem:

① Performance decline on complex feature

② Hard to develop

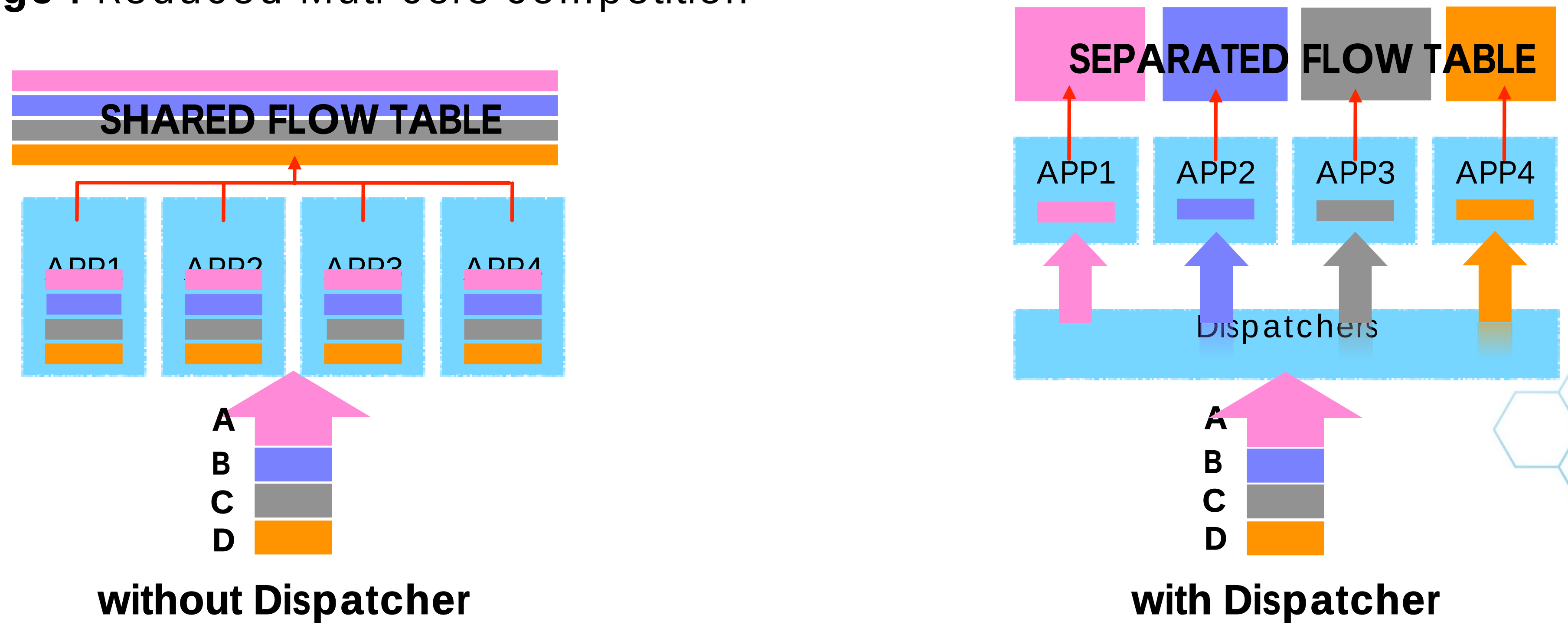
3rd Gen—Dispatcher-Application



1. RSS-binded packets handle
2. 5-tuple hash dispatcher
3. control-plane Vs data-plane

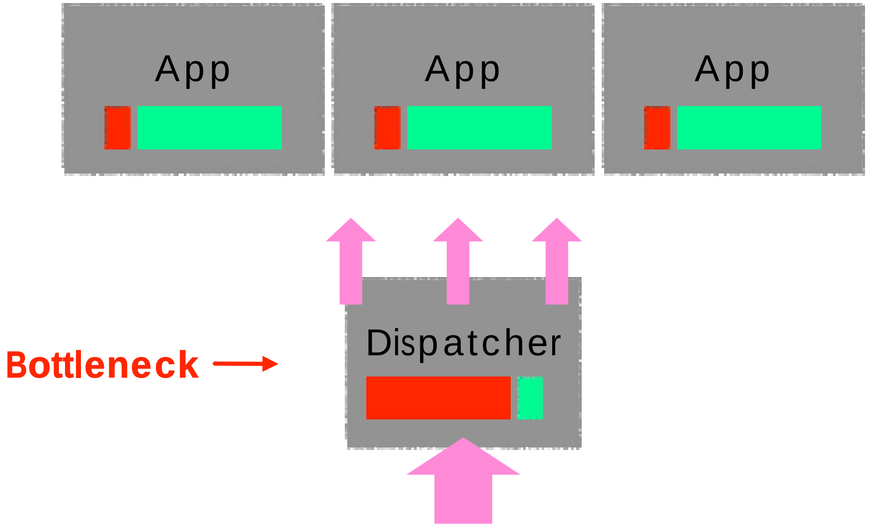
3rd Gen—Dispatcher-Application

Advantage：Reduced Muti-core competition

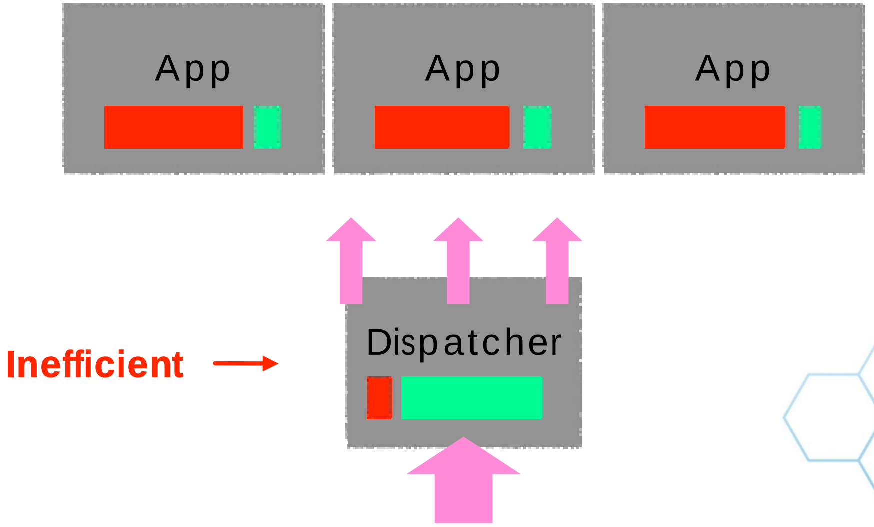


3rd Gen—Dispatcher-Application

Problem：Bottleneck in different situation



Heavy traffic, low complexity

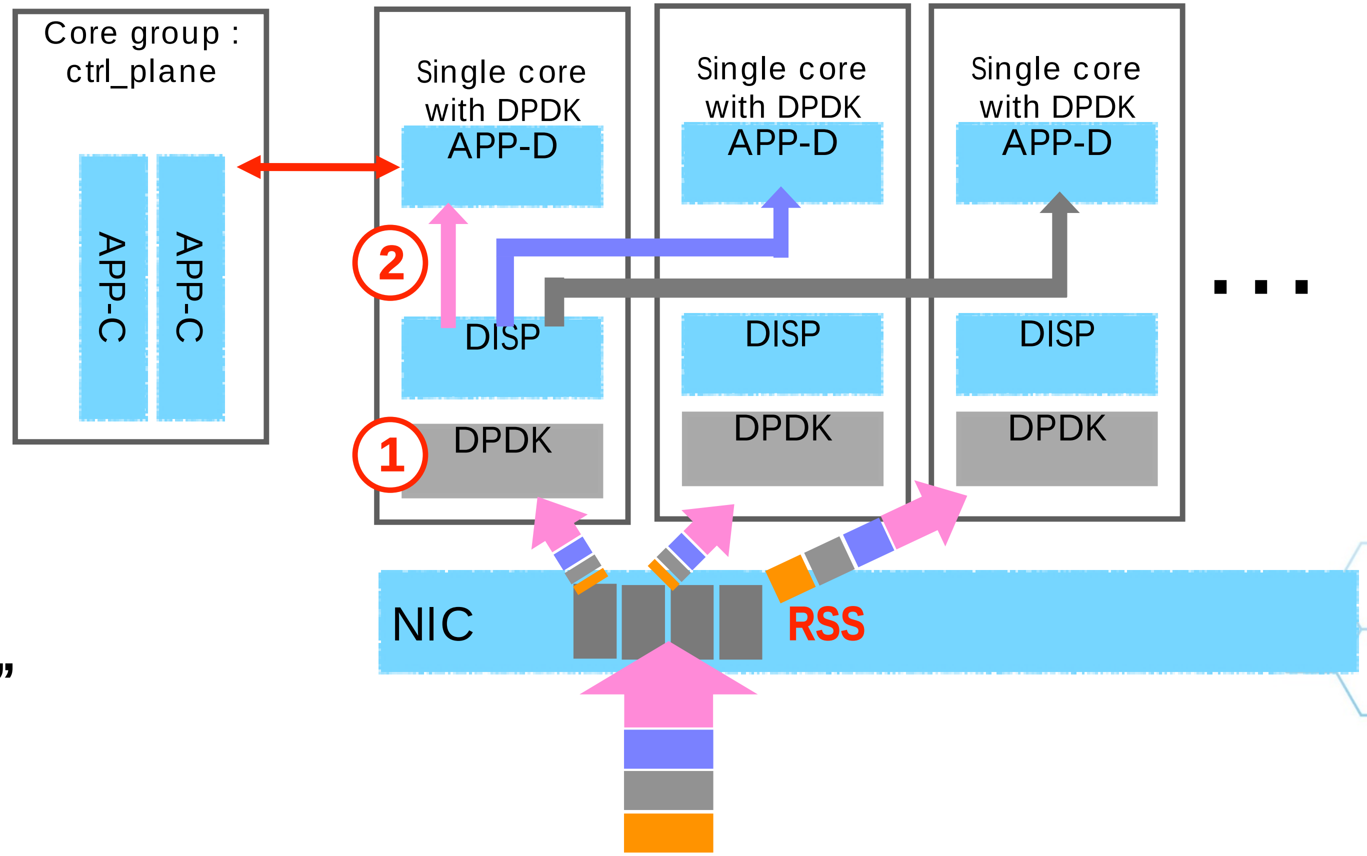


Light traffic, high complexity

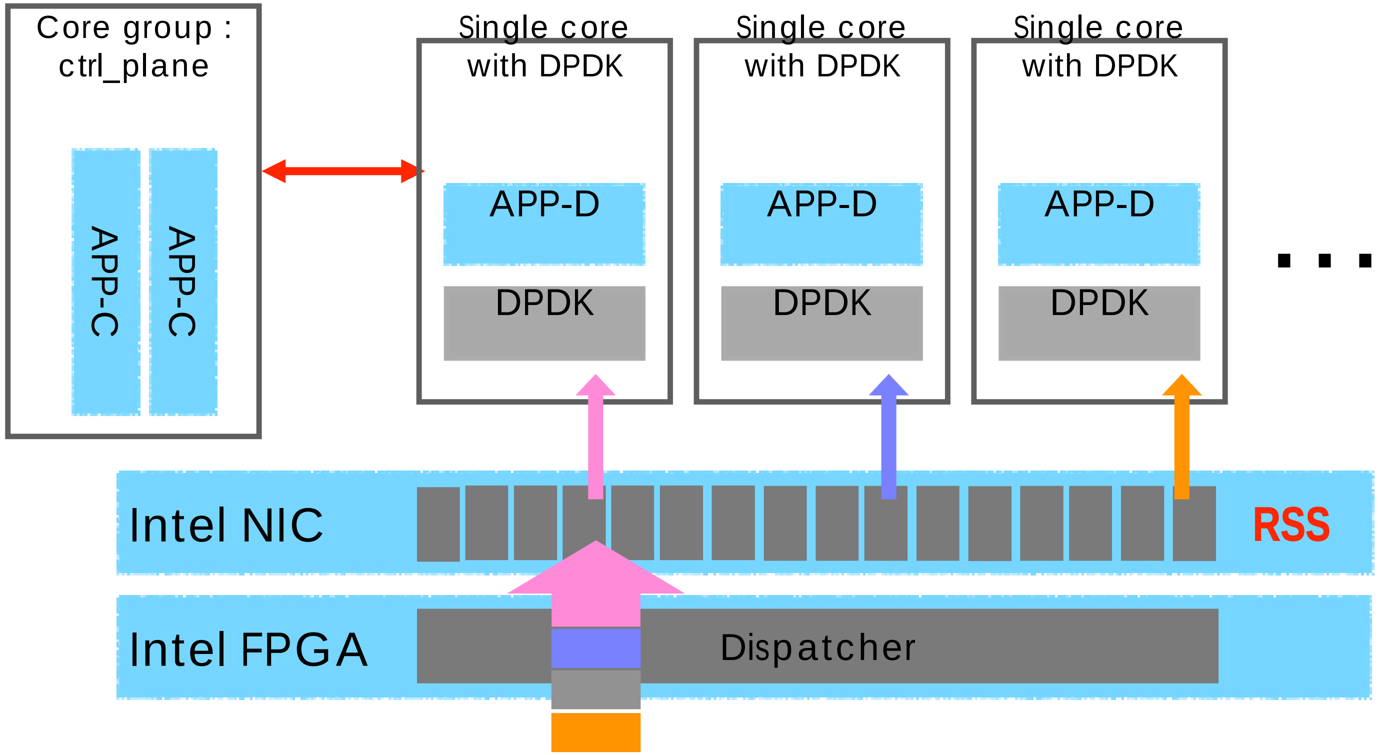
4th Gen—DPDK-equipped system

Two improvement :

1. DPDK-equipped.
2. **“balance-dispatcher”** system**.**



5th Gen—Maybe in the future



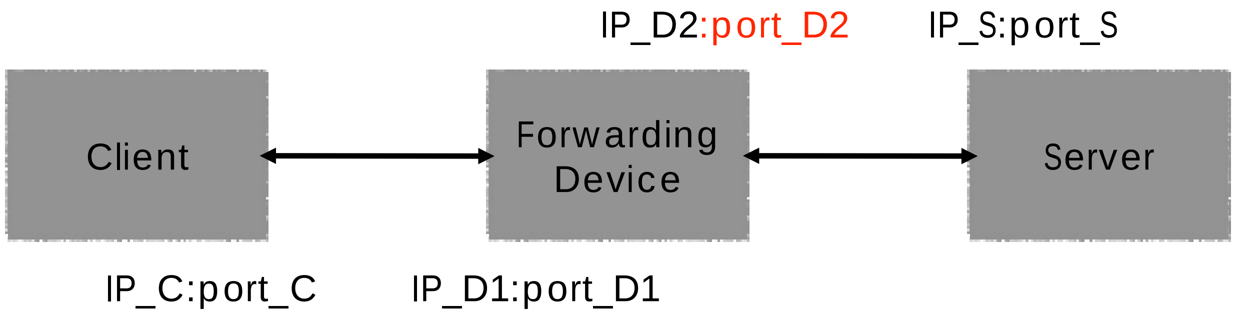
Release CPU cost from dispatcher.

Avoid packets deliver between cores.

More efficient on cache scheduling.

Why dispatcher in software

Can not use RSS hash, why?



Precondition:

1. HASH value of both sides must be consistent
2. port_D2 can be decided

Calculate process :

HASH_VALUE = hash(IP_C, port_C, IP_D1, port_D1)

port_D2 = hash_inverse(HASH_VALUE, IP_D2, IP_S, port_S)

It is difficult to perform a “inverse hash” based on hardware RSS HASH

History of “T1-system”

- 1st Generation —— “kernel driver based” system
- 2nd Generation —— Muti-Core MIPS64
- 3rd Generation —— “Dispatcher-application” system
- 4th Generation —— “Balanced-dispatcher” DPDK-equipped system
- 5th Generation —— “DPDK+FPGA” system
- **Why we need DPDK？How to use DPDK?**

Why DPDK??

DPDK vs netMap

1. **Performance**: E5-2670V3 24cores/1000 policies/64-bytes throughput

	Throughput 64bytes	Latency Average (ns)
Netmap	27 Gbps	43700
DPDK	102.4 Gbps	20601

IxNetwork Report — Run:0001

RFC2544 - Throughput/Latency - Aggregated Results

Trial / Framesize / Iteration	Agg L2 Throughput: AggTx Rate %	Agg L2 Throughput: Agg Rx Rate %	Agg L2 Throughput: Agg Rx Rate FPS	Agg L2 Throughput: Agg Rx Rate Mbps	Agg L1 Throughput: Tx Rate Mbps	Agg L1 Throughput: Rx Rate Mbps	Throughput (frames)	Agg Latency: Min (ns)	Agg Latency: Max (ns)	Agg Latency: Average (ns)
Trial: 1 / FS: 64 / Iter: 7	64.00	64	152380691.7	78018.914			Tx : 1523809520.000 Rx : 1523809309.000 Loss : 211 Loss% : .00	5000	778360	20601.500
Trial: 1 / FS: 512 / Iter: 8	100.00	100	37593857.3	153984.440			Tx : 375939856.000 Rx : 375939856.000 Loss : 0 Loss% : .00	6080	699520	20206.313
Trial: 1 / FS: 1518 / Iter: 8	100.00	100	13003856.9	157918.839			Tx : 130039008.000 Rx : 130039008.000 Loss : 0 Loss% : .00	7920	703280	20329.750

Why DPDK??

DPDK vs netMap

2. **Performance**: CPU cost analysis by oprofiler

51544	18.2802	ipv4_rcv
38557	13.6743	se_resolve_normal_ct.part.19
28482	10.1012	tb_skb_rcv
27258	9.6671	se_ip_conntrack_in
25112	8.9060	tb_nf_hook_slow
11180	3.9650	_recv_raw_pkts_vec
10610	3.7629	tb_skb_send
9838	3.4891	ixgbe_xmit_pkts_vec
9603	3.4057	__se_ip_ct_refresh_acct
8172	2.8982	packet_intercept
7916	2.8074	tb_clear_skb_header
7579	2.6879	jhash_3words
4600	1.6314	tb_stat_flow
4073	1.4445	tb_skb_capture
3683	1.3062	tb_packet_handle_loop
2918	1.0349	tb_rte_memcpy_func.constprop.23
2537	0.8998	tb_flow_stat_policy

DPDK lib

System with DPDK

33951	12.7653	se_resolve_normal_ct.part.19
32165	12.0937	packet_intercept
26541	9.9792	se_ip_conntrack_in
24187	9.0941	nm_send
22004	8.2733	tb_nf_hook_slow
19018	7.1506	nm_recv
11845	4.4536	app_interface_flow_stat_entry
9797	3.6836	__se_ip_ct_refresh_acct
8236	3.0967	tb_clear_skb_header
7830	2.9440	jhash_3words
6416	2.4124	nm_send_skb
6007	2.2586	ipv4_rcv
5821	2.1886	tb_skb_rcv
5576	2.0965	tb_packet_handle_loop
5227	1.9653	tb_skb_xmit
4764	1.7912	tb_stat_flow
3728	1.4017	tb_skb_capture
2864	1.0768	tb_rte_memcpy_func.constprop.23

Netmap lib

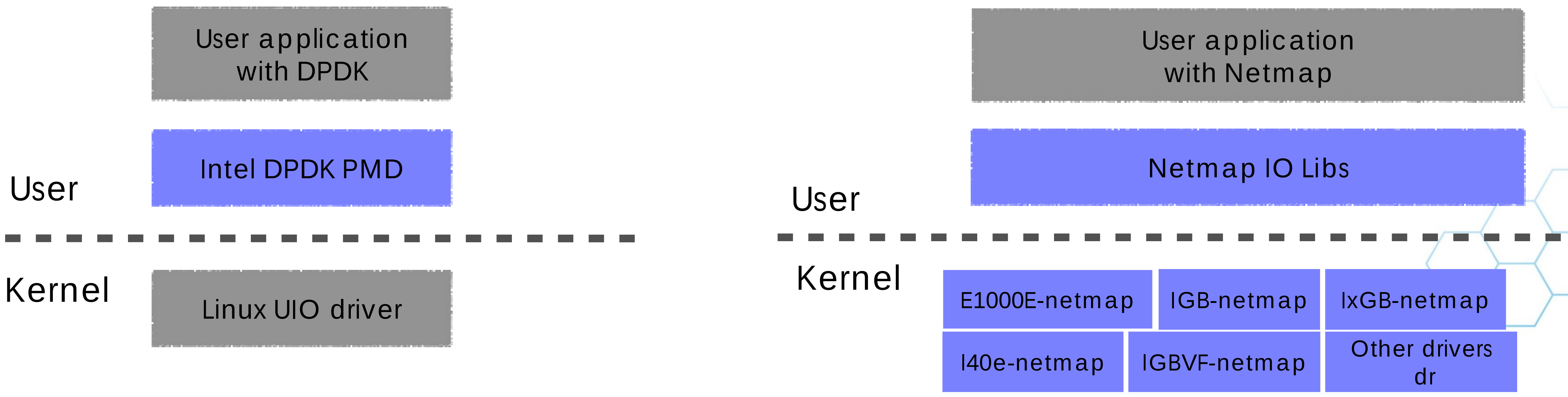
System with Netmap

Why DPDK??

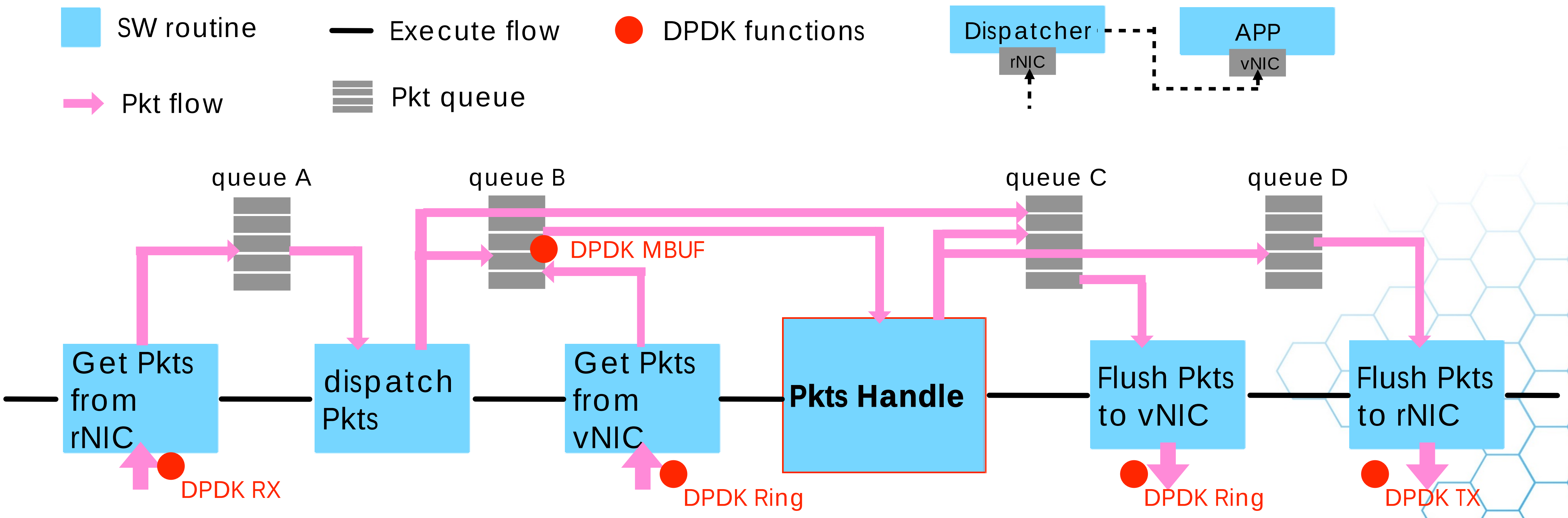
DPDK vs netMap

3. Code maintenance costs

: Code block we should take care of



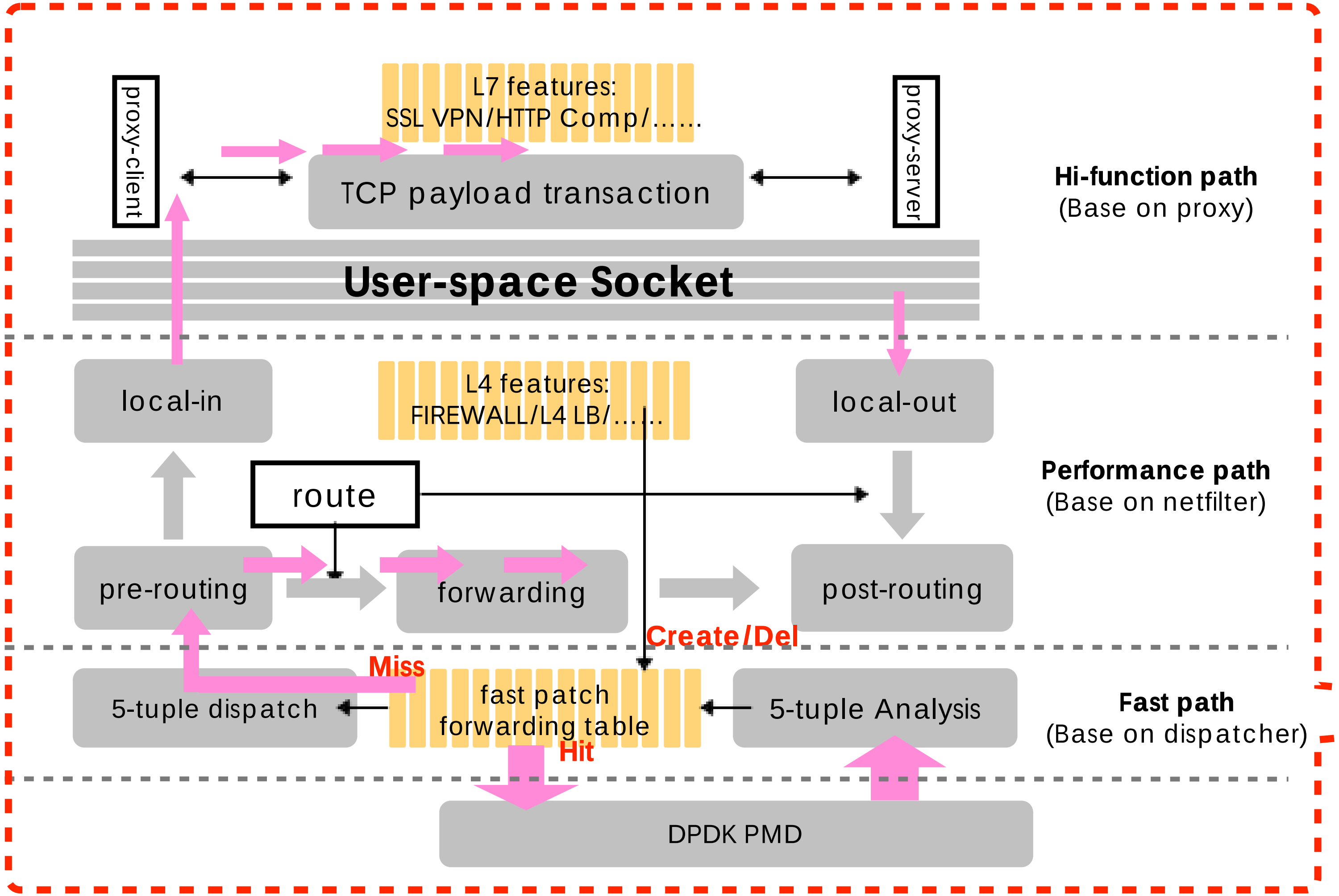
Application with DPDK

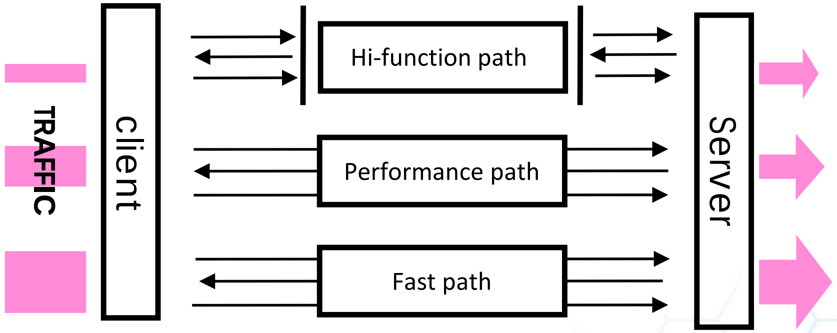


Agenda

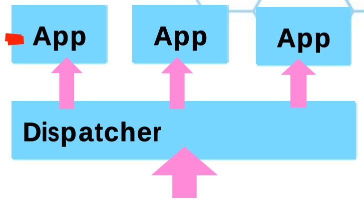
- **Our History —** What is an embedded network device
- **Challenge to us —** Requirements for device today
- **Our solution —** T1 unique embedded network architecture（T1-System）
  - Model of “embedded network architecture”
  - History of T1-system
  - **Business layer of T1-system**
  - An optimization case —— dual-sockets system
  - T1-system as a NFV

Business layer of T1-System: Multi-path traffic handle system





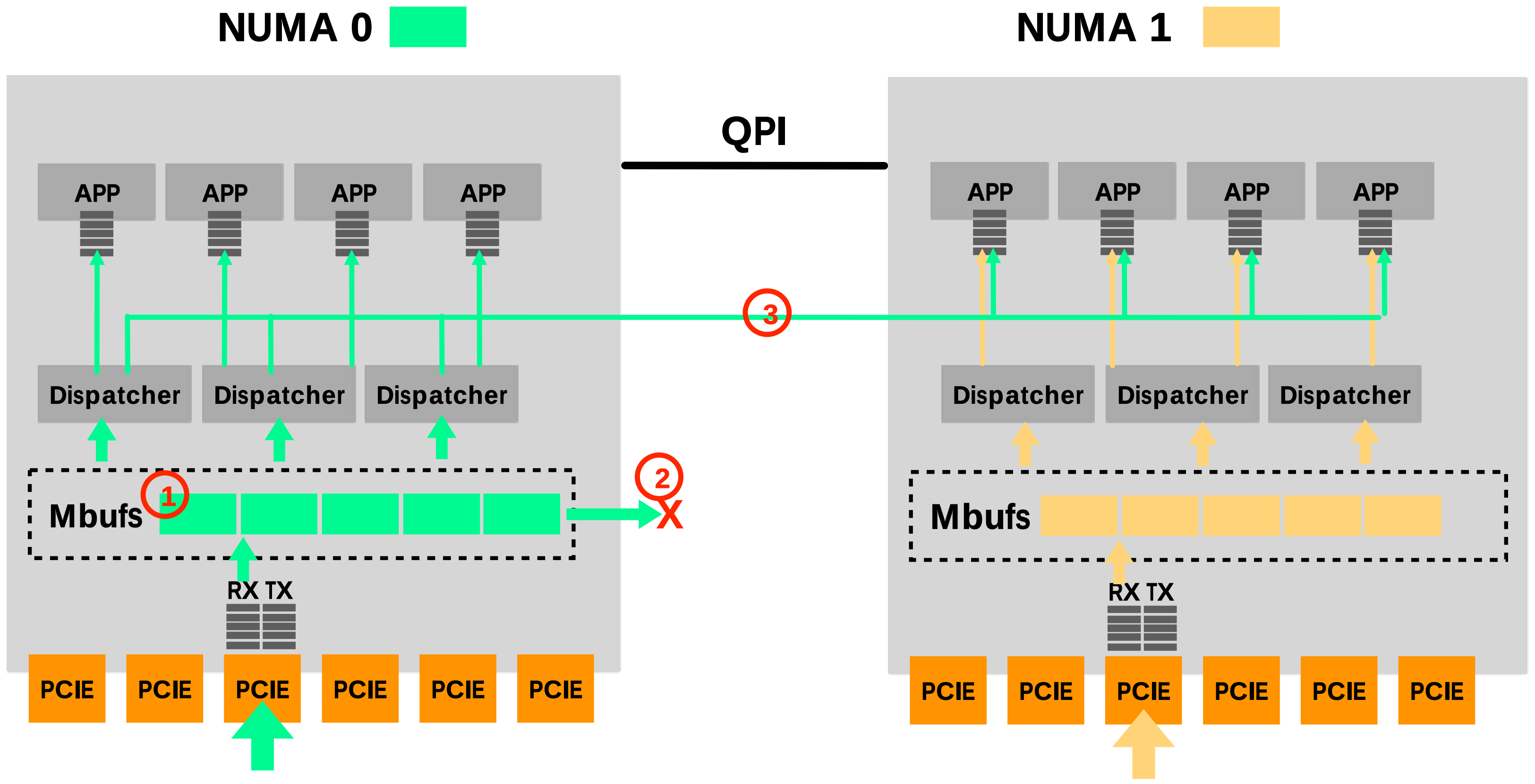
Aim of Multi-path system:
Reduce CPU cost on traffic processing.



Agenda

- **Our History —** What is an embedded network device
- **Challenge to us —** Requirements for device today
- **Our solution —** T1 unique embedded network architecture（T1-System）
  - Model of “embedded network architecture”
  - History of T1-system
  - Business layer of T1-system
  - **An optimization case —— dual-sockets system**
  - T1-system as a NFV

Optimization on Dual-sockets platform



Basic environment :

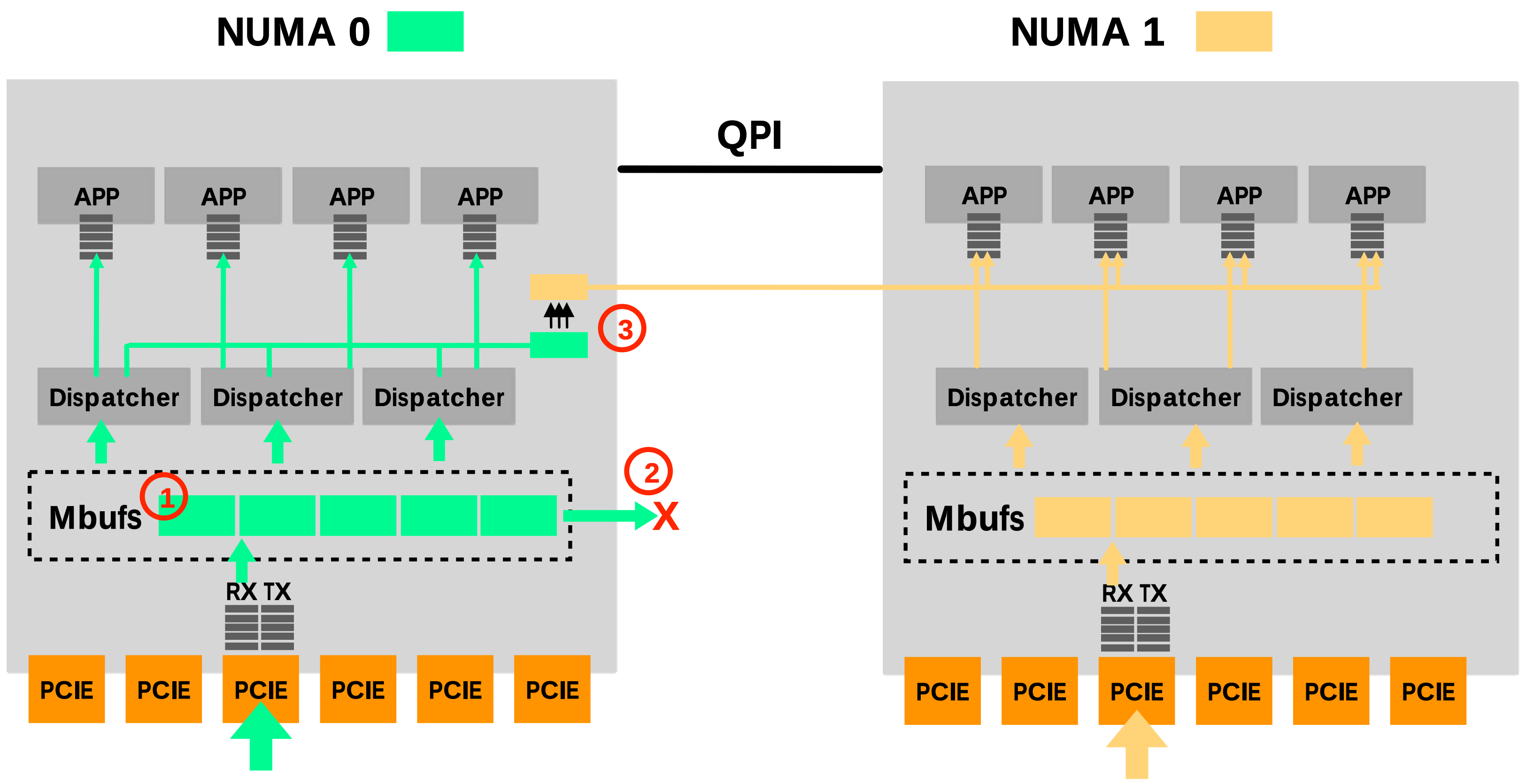
① Separated buffers and queues initialization on each Numa node

② Ethernet ports bind with a single-node.

Case 1 : Packets cross-QPI

③ In case of simple handle of packets, such as IP forwarding.

Optimization on Dual-sockets platform



Basic environment：

1. Separated buffers and queues initialization on each Numa node
2. Ethernet ports bind with a single-node.

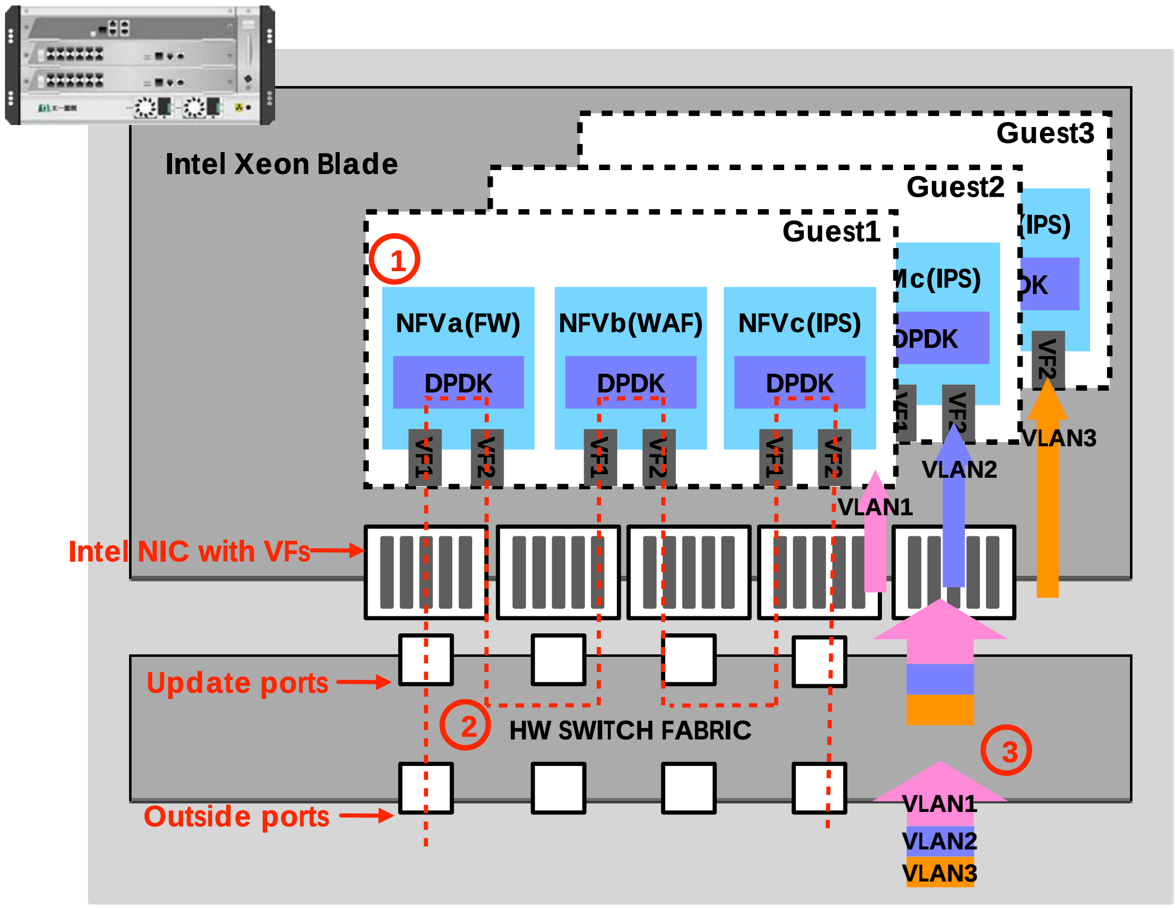
Case 2： Packets copy mode

3. In case of complex handle of packets, such as traffic audit.

Agenda

- **Our History —** What is an embedded network device
- **Challenge to us —** Requirements for device today
- **Our solution —** T1 unique embedded network architecture（T1-System）
  - Model of “embedded network architecture”
  - History of T1-system
  - Business layer of T1-system
  - An optimization case —— dual-sockets system
  - **T1-system as a NFV**
    - NFV resource pool
    - Fusion gateway
    - New solution: OVS with DPDK

NFV Case1:NFV Resource pool



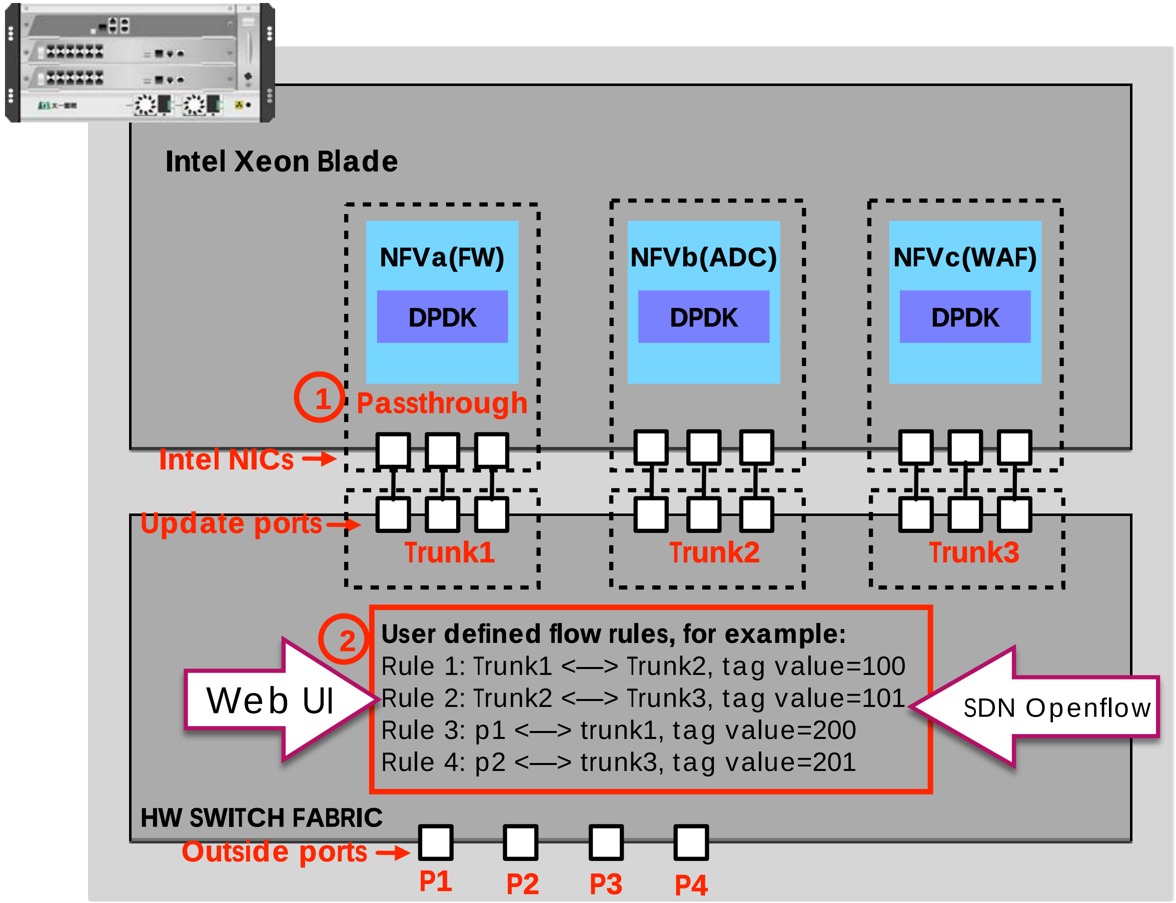
NFV Resource pool :

1. Multiple NFVs for each guest
2. Traffic between NFVs in the same guest is forwarding by HW switch fabric
3. Traffic is isolated by vlan tag between guests

scene :

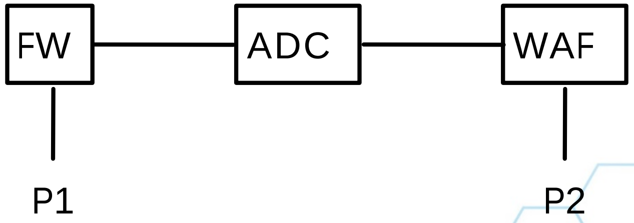
Multi-tenant in data-center/ same flow-define template for each tenant/Elastic expansion

NFV Case2:Fusion gateway



Fusion gateway:

1. Passthrough mode for IO Virtualization
2. Flexible flow-define rules:



scene：

Gateway position/Face to network/High performance/Feature fusion

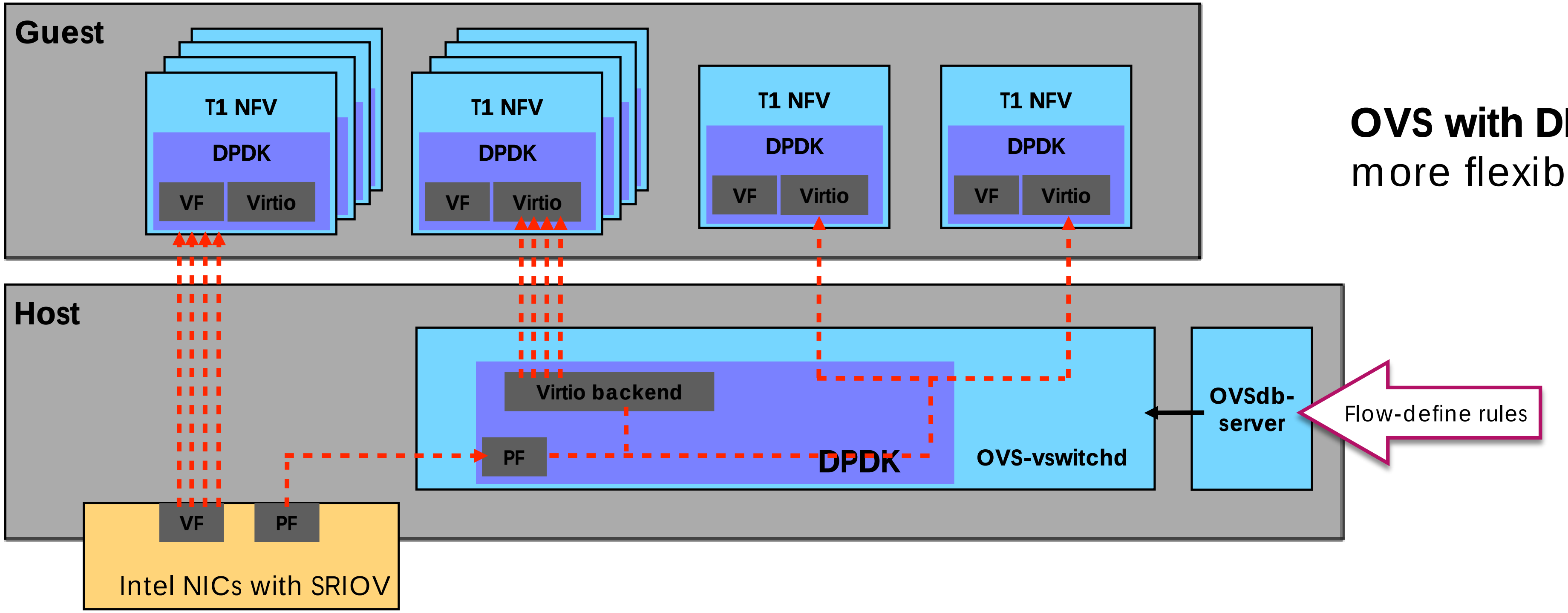
About NFV-Comparison

Comparison of two scenarios

	IO Virtualization	Face to	performance requirement	number of VMs	Configuration focus
NFV resource pool	VF(SR-IOV)	Guest	Low	High	Virtual machine management
Fusion gateway	Passthrough	Network	High	Low	flow-define rules configuration

Limitation：Rely on Hardware fabric

New solution —OVS with DPDK



OVS with DPDK is a low cost, more flexible alternative.

Thank you!