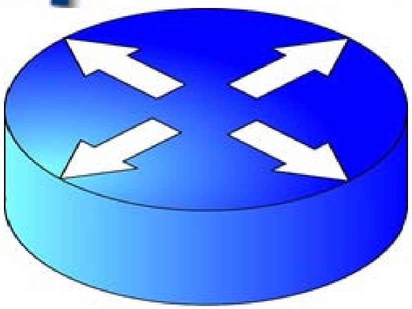
Software Defined Networking

Network Functions Virtualisation

David Hausheer, Leonhard Nobach

Department of Electrical Engineering
and Information Technology
Technische Universität Darmstadt

E-Mail: hausheer@ps.tu-darmstadt.de
http://www.ps.tu-darmstadt.de/teaching/sdn



*Based on original slides by Panagiotis Georgopoulos (ETH Zurich)

Network Functions Virtualisation

Credits: Panagiotis Georgopoulos (ETH Zurich)

Motivation - Problem Statement behind NFV

- Complex carrier networks
  - with a large variety of proprietary nodes and hardware appliances

- Launching new services is difficult and takes too long
  - Space and power to accommodate
  - requires just another set of boxes (i.e. hw) which needs to be integrated

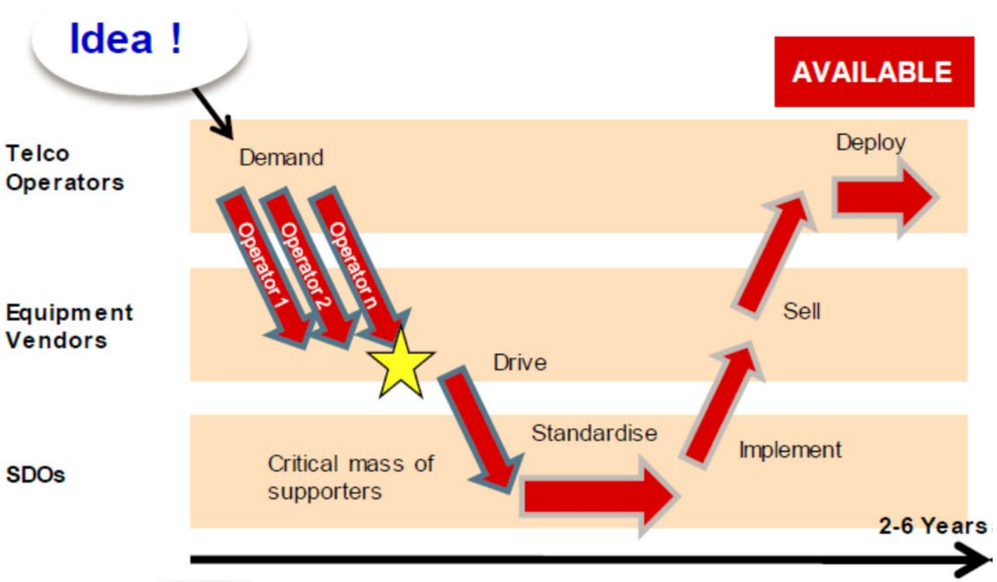
- Operation is expensive
  - Rapidly reach end of life due to existing procure-design, & integrate-deploy cycle

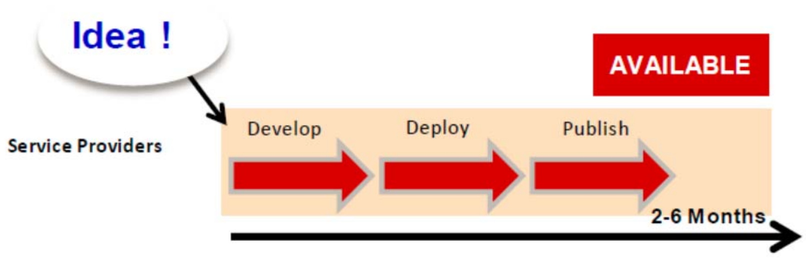
Traditional Network model

- Network functionalities are **based on specific HW&SW**
- **One physical node per role**

Innovation Cycle

- Telco Cycle
- Service Providers Cycle

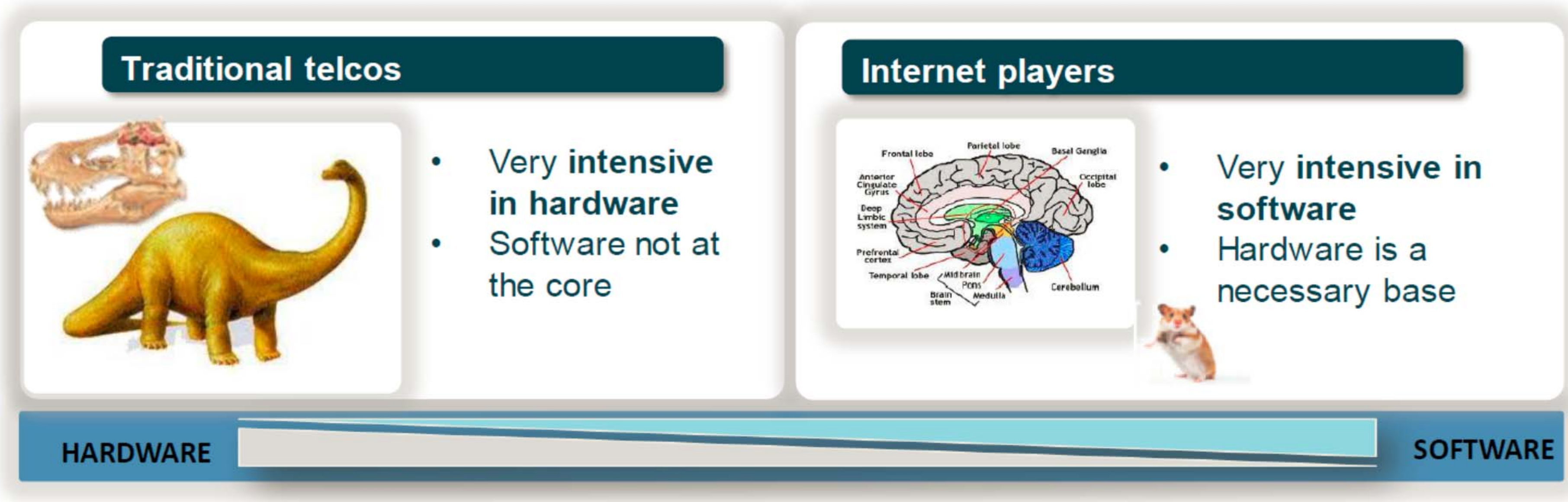




2-6 years

2-6 months

Enter the Software-Defined Era



AT&T, Telefonica,
Telebras

Google, Facebook

Adapt to survive: Telco evolution focuses on shifting from hardware to software

Source: Adapted from D. Lopez Telefonica I+D, NFV

What is the Future of Networking?

TECHNISCHE UNIVERSITÄT DARMSTADT

Trends

- **High performance** servers shipped in very high volume
- **Cloud** services
- **Mobility**; explosion of devices and traffic
- **Software-defined networking**
- New **virtualization technologies** that abstract underlying hardware yielding elasticity, scalability and automation
- **Convergence** of computing, storage and networks

Challenges

- Huge **capital investment** to deal with current trends
- Network operators face an increasing **disparity between costs and revenues**
- **Complexity**: large and increasing variety of proprietary hardware appliances in operator's network
- Reduced **hardware lifecycles**
- **Lack of flexibility and agility**: cannot move network resources where & when needed
- **Launching new services is difficult and takes too long**. Often requires yet another proprietary box which needs to be integrated

Source: Adapted from D. Lopez Telefonica I+D, NFV

Network Functions Virtulisation is / provides:

- The means to implement network functions in software that today run on proprietary network hardware, leveraging high performance servers and IT virtualization

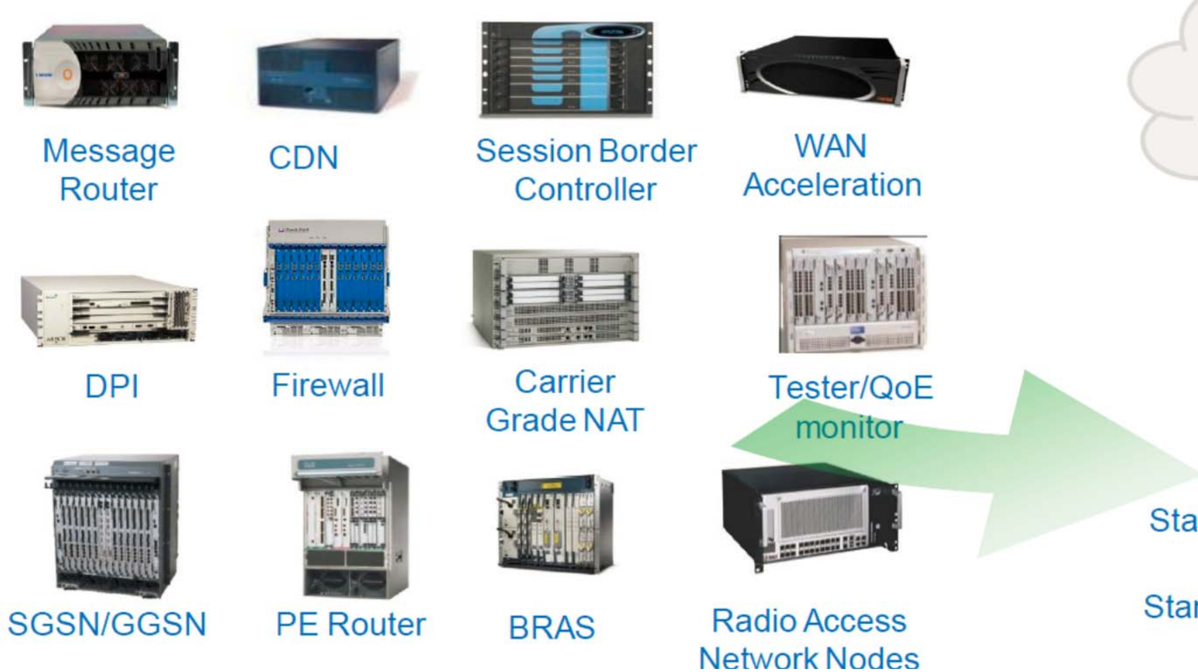
- A way to make the network more flexible and simple by minimising dependence on HW constraints

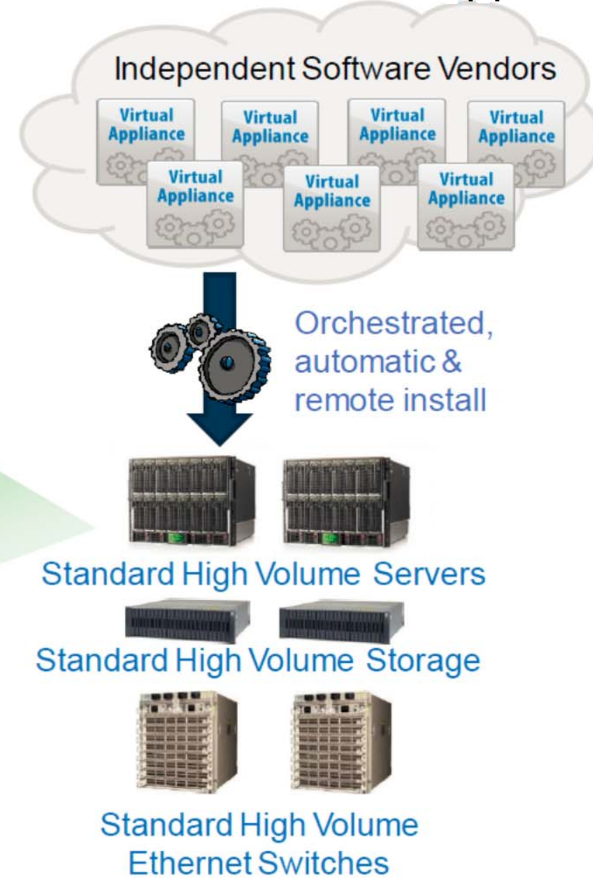
- A helpful analogy :
  - SDN is lifting off the control plane of hardware networking devices to a centralised entity
  - NFV is lifting off the functions/services of proprietary hardware to software components in standard servers/cloud servers

NFV Target

Classical Network Appliance Approach

Network Virtualisation Approach

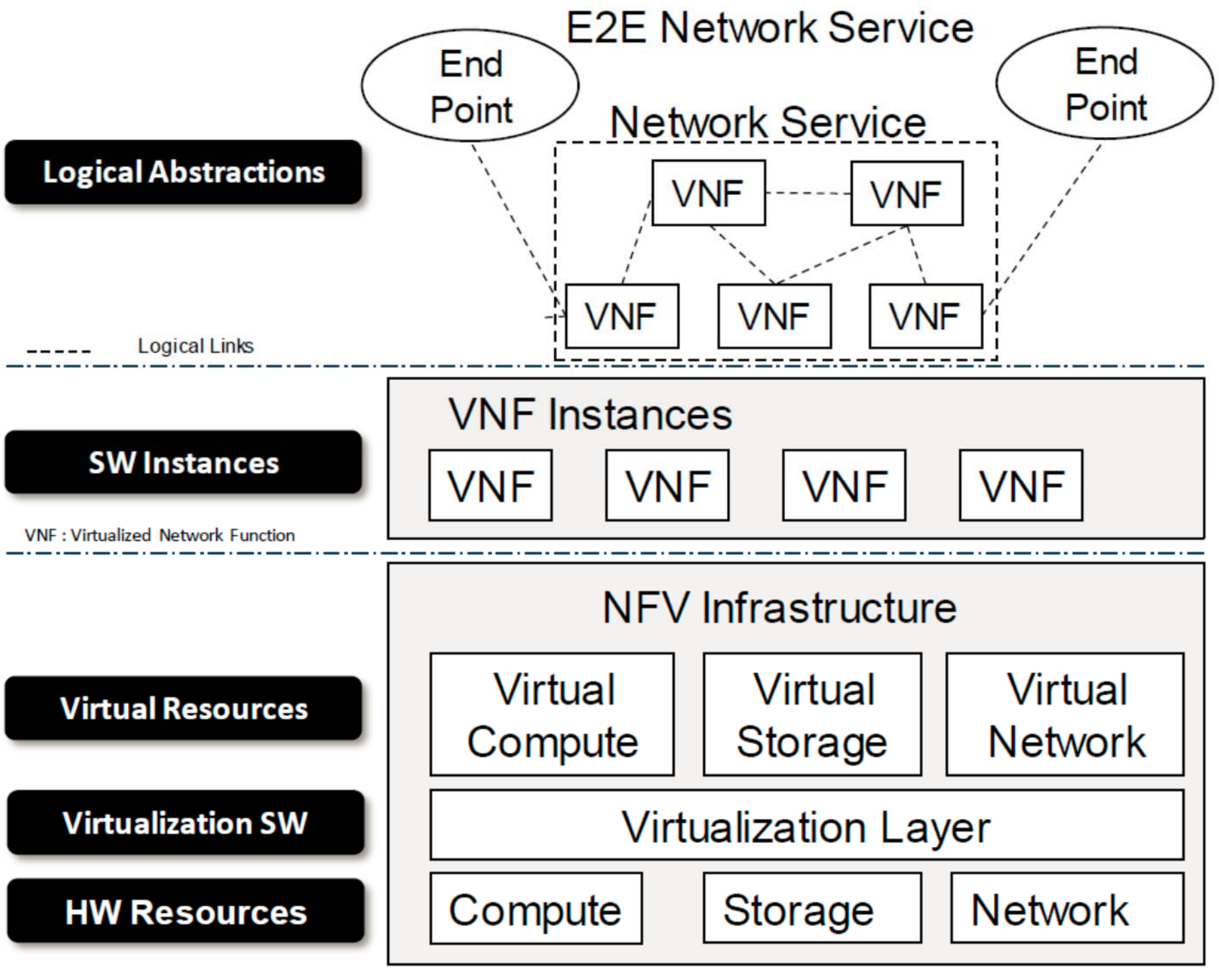




- Fragmented non-commodity hardware
- Physical install per appliance per site
- Hardware development large barrier to entry for new vendors, constraining innovation & competition

Source: Adapted from NFV

NFV Architecture based on Layers



Source: Adapted from D. Lopez Telefonica I+D, NFV

Network Functions Virtualization Aims to…

- Support multi-versioning and multi-tenancy of network functions, which allows use of a single physical platform for different applications, users and tenants
- Enable new ways to implement resilience, service assurance, test and diagnostics and security surveillance
- Provide opportunities for pure software players
- Facilitate innovation towards new network functions and services that are only practical in a pure software network environment
- Applicable to any data plane packet processing and control plane functions, in fixed or mobile networks
- **NFV aims to ultimately transform the way network operators architect and operate their networks, but change can be incremental**

Source: Adapted from D. Lopez Telefonica I+D, NFV

Network Functions Virtualization Aims to…

TECHNISCHE UNIVERSITÄT DARMSTADT

BUT

- NFV will only scale :
  - If management and configuration of functions can be automated
  - If placement of software functions and steer of network traffic can be automated

Benefits & Promises of NFV

- Reduced equipment costs (CAPEX) through consolidating equipment and economies of scale of IT industry
- Reduced (OPEX) operational costs: reduced power, reduced space, improved network monitoring
- Software-oriented innovation to rapidly prototype and test new services
  - Flexibility to easily, rapidly, dynamically provision and instantiate new services in various locations
- Increased speed of time to market
  - by minimising the typical network operator cycle of innovation

Source: NFV

Benefits & Promises of NFV

TECHNISCHE UNIVERSITÄT DARMSTADT

- Improved operational efficiency
  - by taking advantage of the higher uniformity of the physical network platform and its homogeneity to other support platforms.
- Availability of network appliance multi-version and multi-tenancy, allows a single platform for different applications, users and tenants.
- More service differentiation & customization
- Enables a variety of eco-systems and encourages openness
- Encouraging innovation to bring new services and generate new revenue streams

Source: Adapted from D. Lopez Telefonica I+D, NFV

So Telco Operators can achieve

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Manage thousands of devices
3. **Programmable**: Should be able to change behavior on the fly
4. **Dynamic Scaling**: Should be able to change size, quantity
5. **Automation**
6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
8. **Multi-tenancy**
9. **Service Integration**
10. **Openness**: Full choice of modular plug-ins

Note: These are very similar reasons as to why we need SDN

Source: Adapted from Raj Jan

Some NFV Use Case Examples

...not in any particular order

- Switching elements: BNG, CG-NAT, routers
- Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW
- Home networks: Functions contained in home routers and set top boxes to create virtualised home environments
- Tunnelling gateway elements: IPSec/SSL VPN gateways
- Traffic analysis: DPI, QoE measurement
- Service Assurance: SLA monitoring, Test and Diagnostics
- NGN signalling: SBCs, IMS
- Converged and network-wide functions: AAA servers, policy control and charging platforms
- Application-level optimisation: CDNs, Cache Servers, Load Balancers, Application Accelerators
- Security functions: Firewalls, virus scanners, intrusion detection systems, spam protection

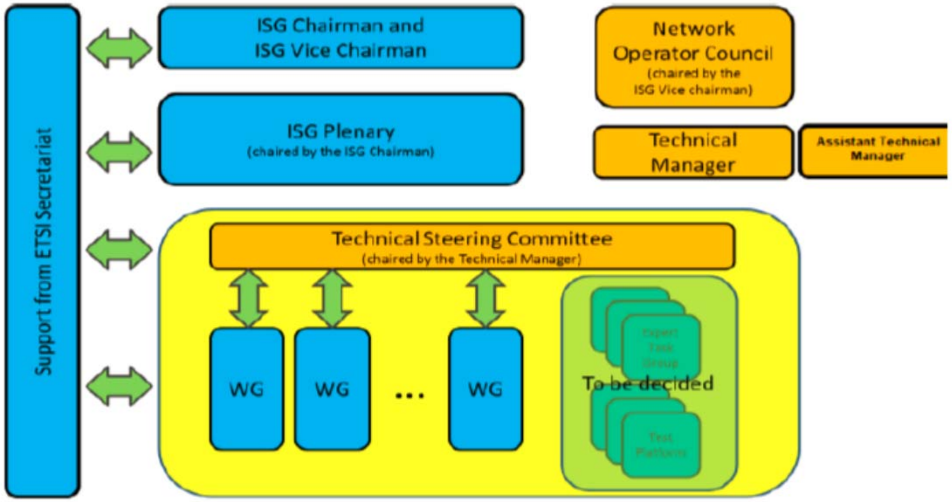
Source: NFV

The ETSI NFV ISG

- ETSI: European Telecommunications Standards Institute
- Global operators-led Industry Specification Group (ISG) under the auspices of ETSI
  - ~150 member organisations
- Open membership
  - ETSI members sign the "Member Agreement"
  - Non-ETSI members sign the "Participant Agreement"
  - Opening up to academia
- Operates by consensus
  - Formal voting only when required
- Deliverables: White papers addressing challenges and operator requirements, as input to SDOs
  - Not a standardisation body by itself

Currently, four WGs and two EGs

- Infrastructure
- Software Architecture
- Management & Orchestration
- Reliability & Availability
- Performance & Portability
- Security



Source: NFV

NFV Goals and SDN Goals

TECHNISCHE UNIVERSITÄT DARMSTADT

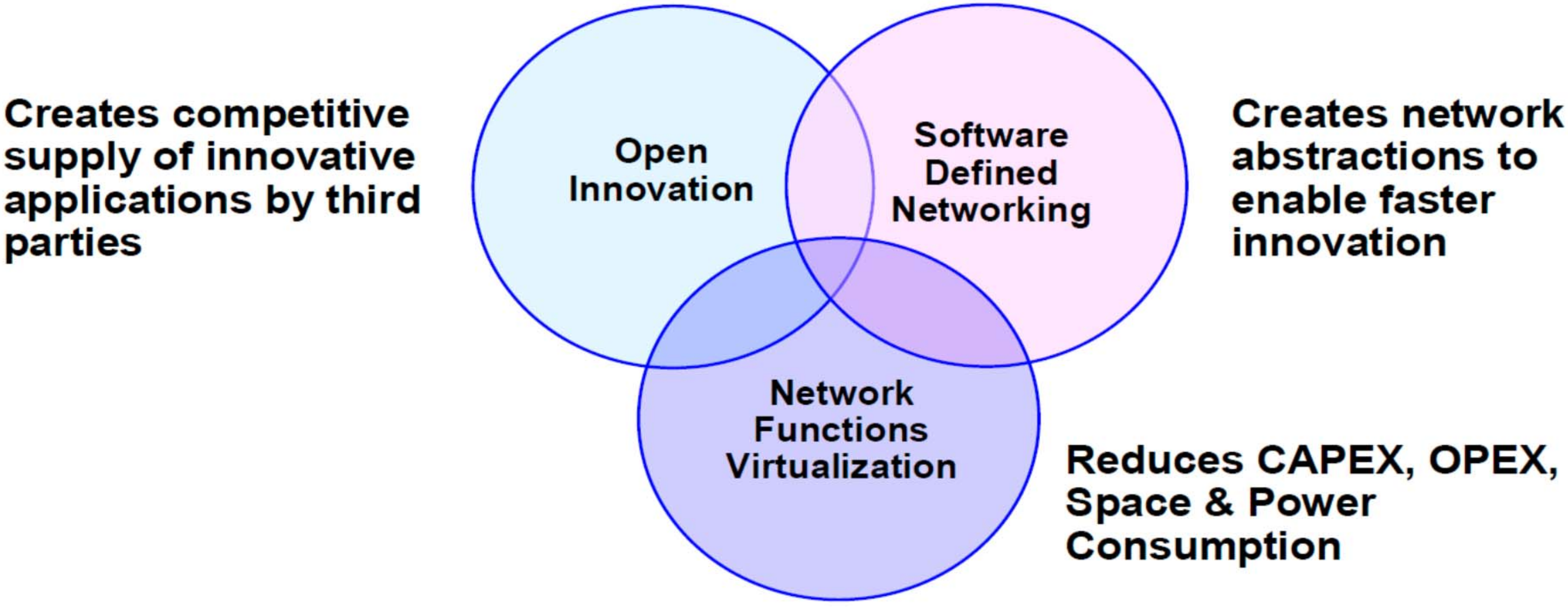
- **NFV**: re-definition of network equipment architecture
- NFV was born to meet Telco/Service Provider needs:
  - Implement network functions in software and remove HW dependencies
  - Lower CAPEX by reducing/eliminating proprietary hardware
  - Consolidate multiple network functions onto industry standard platforms

- **SDN**: re-definition of network architecture
- SDN comes from the academia/IT world:
  - Separate the data and control planes & centralize control
  - Deliver the ability to program network behavior using well-defined interfaces

Source: NFV

NFV and SDN

- NFV and SDN are highly complementary
- Both efforts are mutually beneficial but not dependent on each other



Source: NFV

NFV and Performance

Leonhard Nobach

NFV and Performance

- Challenges
  - Virtualization overhead
    - Virtualization technology: Higher latency and reduced bandwidth
  - Compete with hardware acceleration
    - Appliances use ASICs, NPUs and FPGAs
    - Often achieve line rate
- Solution Approaches
  - **Scale-Out**
    - common mitigation, but requires more hardware, rack space and energy.
  - Reduction of Virtualization and I/O **Overhead**
  - **Hardware** Acceleration

ClickOS

- Effort to achieve near-line-rate performance with NFV
  - On general purpose processor (x86) platforms (no AH)
- Xen-based
  - Better performance than KVM claimed (questionable)
- Idea: Reduce virtualization overhead by various **optimizations**
  - Usage of software switch VALE/Netmap
  - Reduction of VM entries/exits
  - Usage of poll-mode drivers (PMD) and Interrupt Coalescing
- Results (64 byte packets, low-end servers)
  - **45 µs delay** (KVM: 65µs)
  - **7-8 M packets per second** (KVM/Xen out of the box: 0,8 Mpps)
  - 8Mpps * 64 Bytes/Packet * 8 Bits/Byte => **4096 Mbit/s**

[1] J. Martins et al. USENIX NSDI 2014.

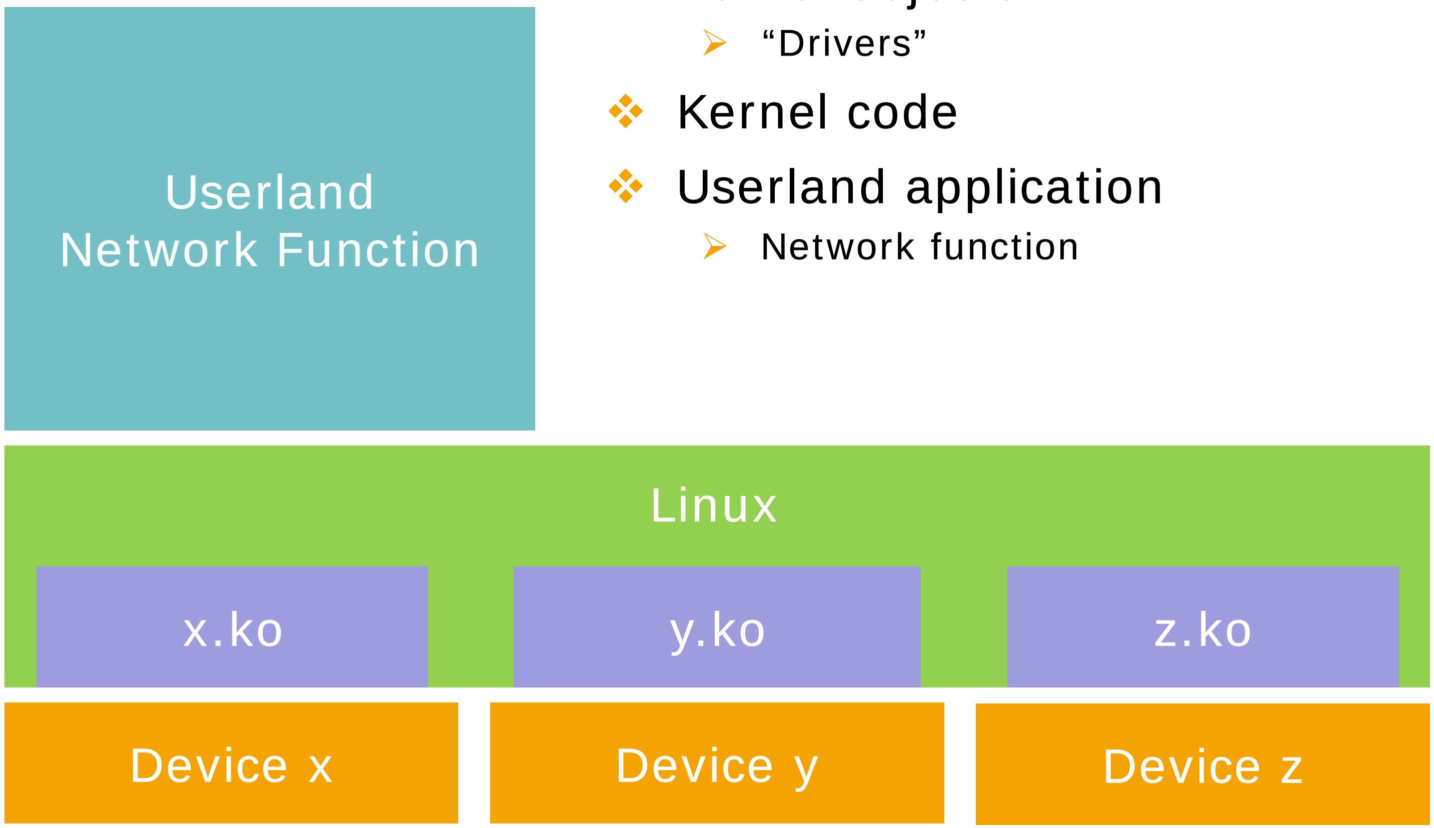
Data Plane Development Kit (DPDK)

DPDK - Overview

- Developed by Intel (Reference: http://dpdk.org/)
- Set of **drivers / libraries** to speed up network I/O processing
- Only works with compatible hardware
- Library / driver components:
  - Queue handling
  - Packet classification
  - Poll-mode drivers (PMD)
- Performance results differ (64 byte packets, Intel Xeon)
  - **Intel*: 80 Mpps** (Workload: Unknown, maybe **bare throughput**)
  - **Pongrácz et al. [2]: <11 Mpps** (Workload: OpenFlow Software Switch)
  - **Emmerich et al. [3]: <10 Mpps** (Workload: DPDK Open vSwitch)
- **Results very dependent of network function!**

*http://www.intel.com/content/www/us/en/communications/communications-packet-processing-brief.html

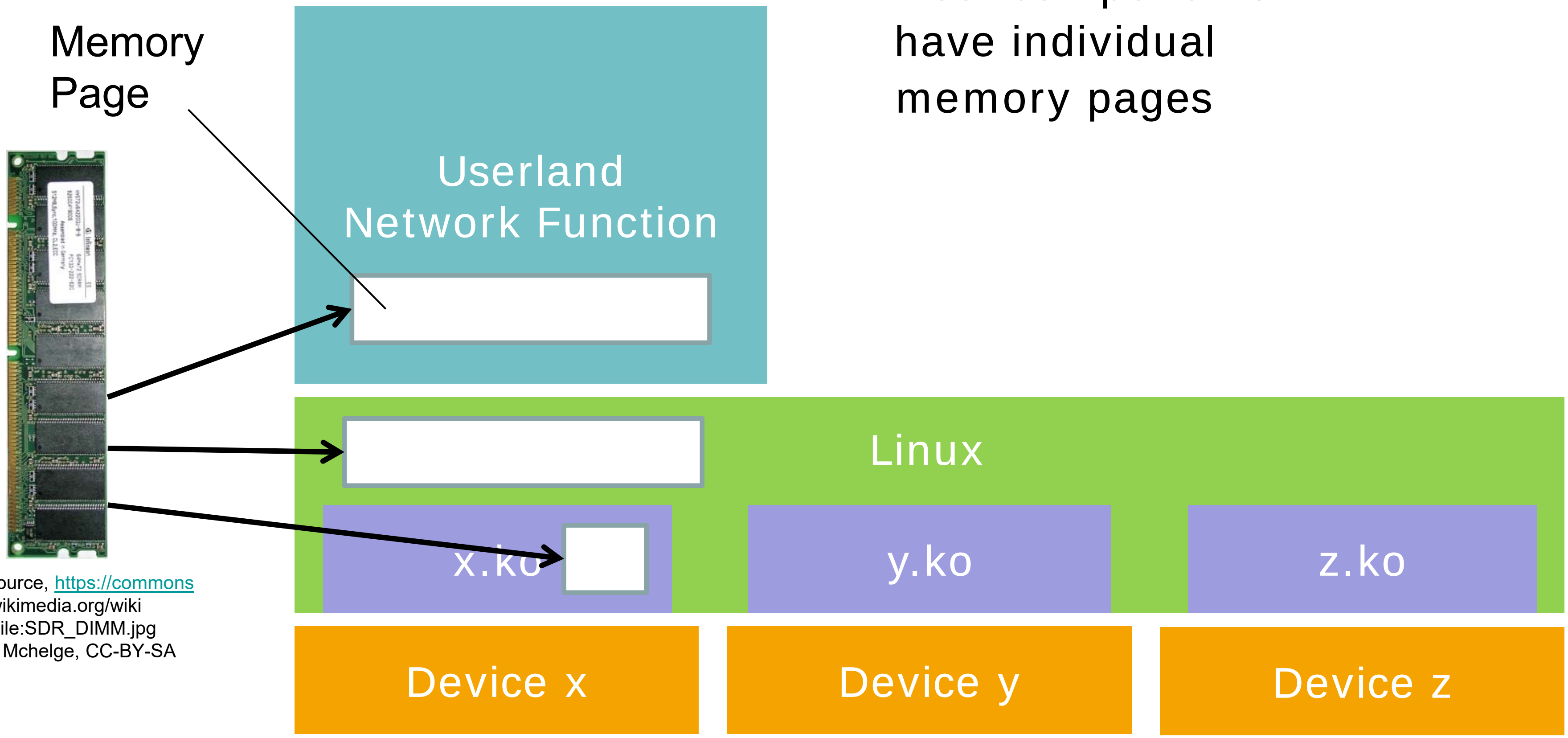
Common Linux Network Architecture



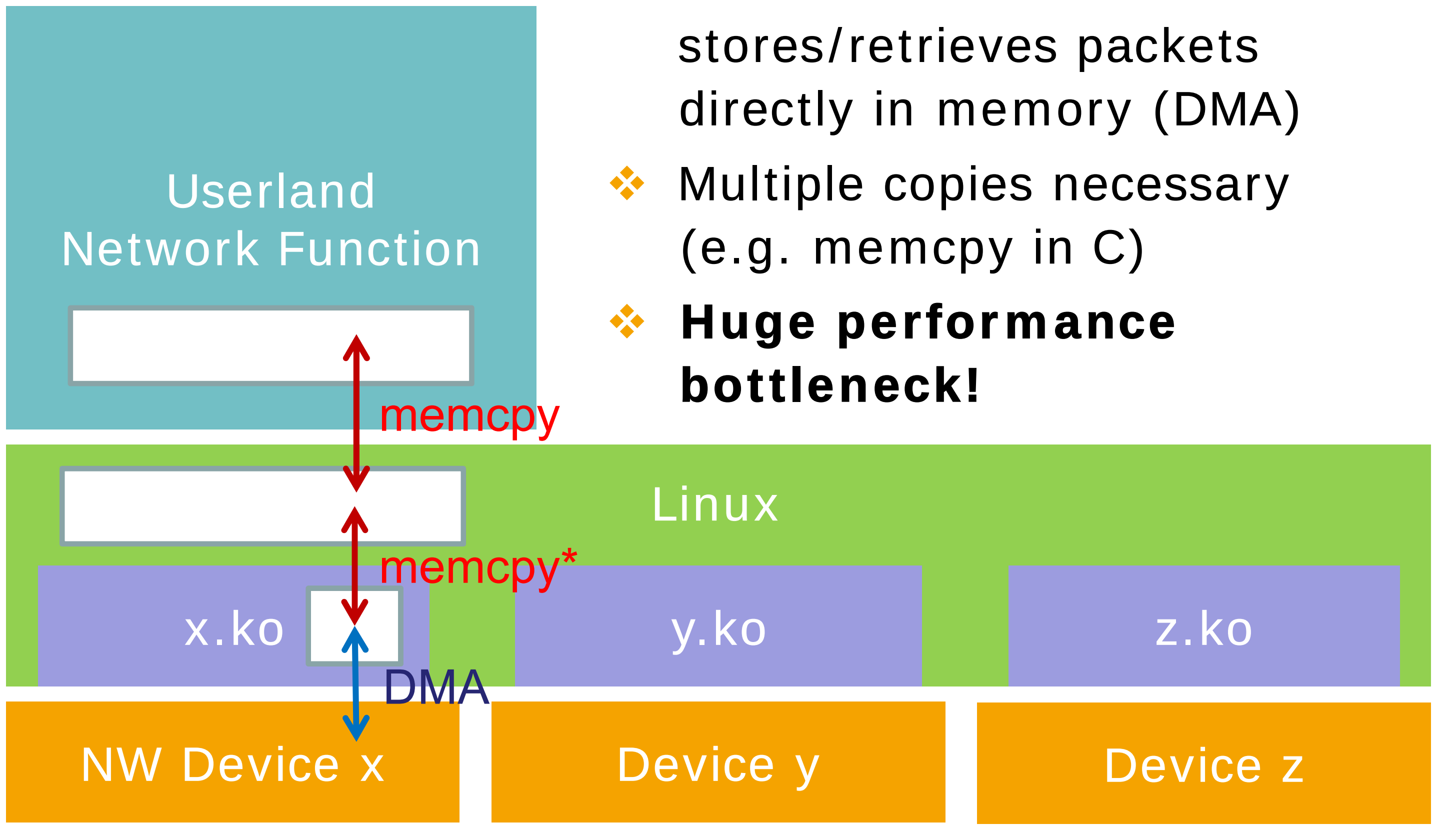
- Kernel objects
  - “Drivers”
- Kernel code
- Userland application
  - Network function

Common Linux Network Arch. (2)

- Most components have individual memory pages



Common Linux Network Arch. (3)

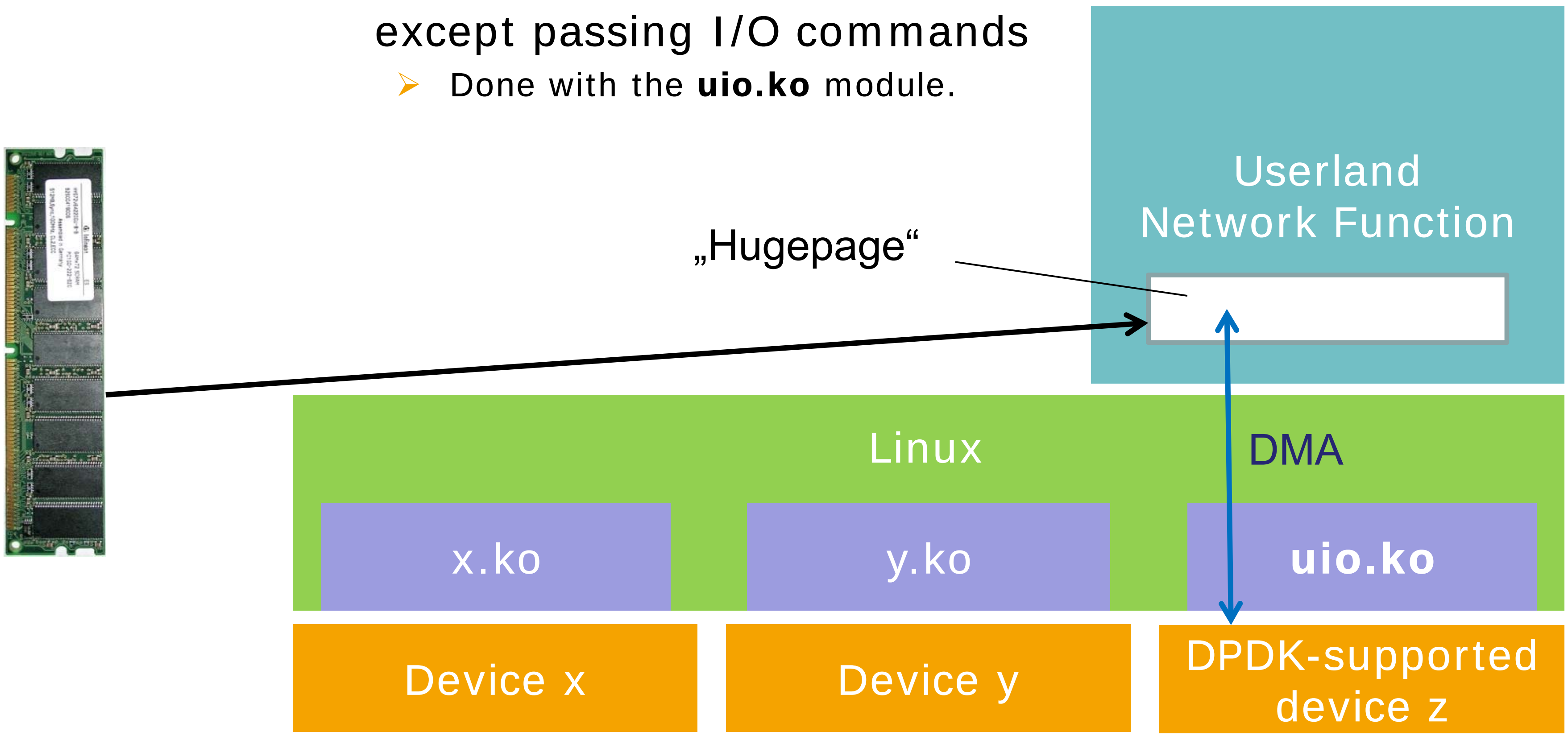


- Network interface stores/retrieves packets directly in memory (DMA)
- Multiple copies necessary (e.g. memcpy in C)
- **Huge performance bottleneck!**

*It depends on your configuration if this memcpy occurs. Even more than two memcpies are possible.

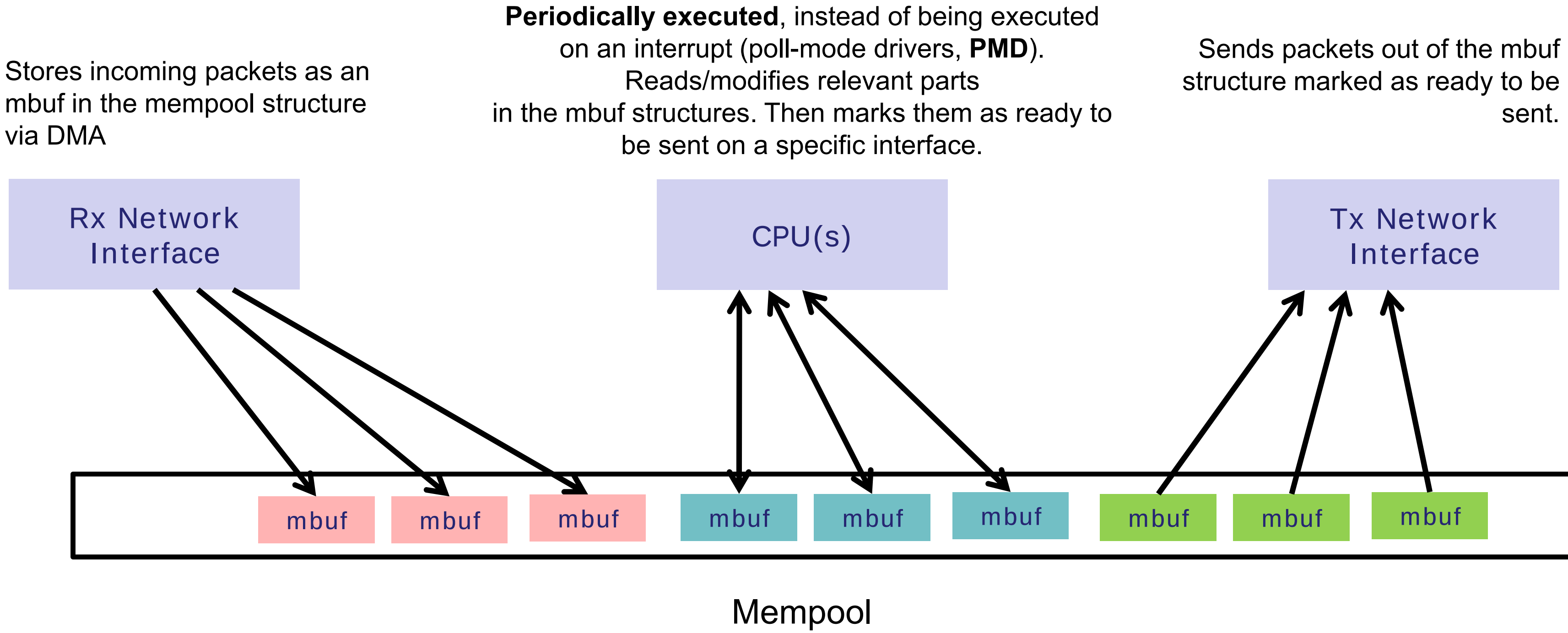
DPDK Architecture

- DPDK: Kernel not involved, except passing I/O commands
  - Done with the **uio.ko** module.



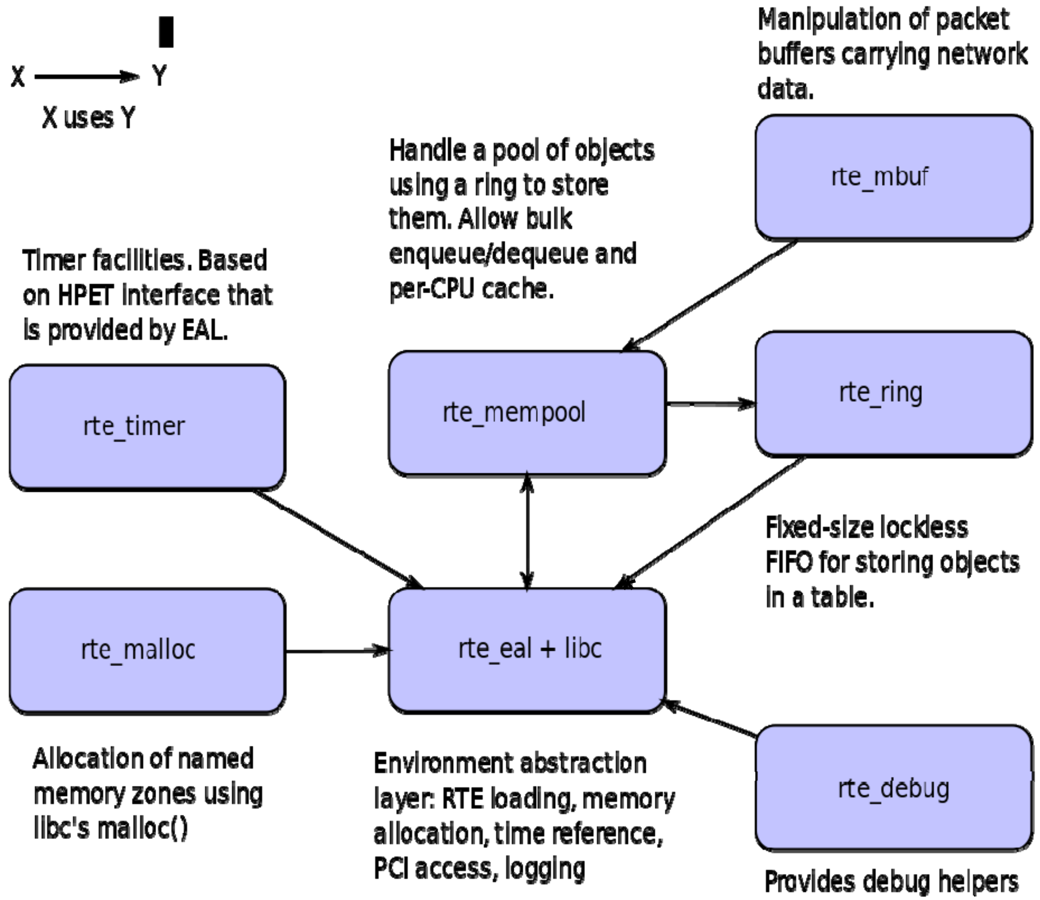
DPDK – Most Important Data Structures

- **rte_mempool:** Memory structure commonly used to store packets (rte_mbuf structures)
- **rte_mbuf:** Structure containing a single packet



Programming with DPDK

- **Note:** This prepares you for the upcoming labs.



Source: http://dpdk.org/doc/guides/prog_guide/overview.html#memory-pool-manager-librte-mempool

DPDK, RTE Minimalistic Init Process

```
rte_eal_init(argc, argv);

rte_eth_dev_configure(ETHDEV_ID, 1, 1, &default_ethconf);

mempool = rte_mempool_create(…);

rte_eth_rx_queue_setup(ETHDEV_ID, 0, … &rx_conf, mempool);

rte_eth_tx_queue_setup(ETHDEV_ID, 0, …, &tx_conf);

rte_eth_dev_start(ETHDEV_ID)

rte_eal_remote_launch(do_dataplane_job, NULL, our_lcore);
```

Pointer to a function to call on a dedicated core

DPDK, Packet Manipulation

```
static int do_dataplane_job(__attribute__((unused)) void *dummy) {
while(1) {

          uint32_t rx_pkt_count = rte_eth_rx_burst(ETHD
          int i;

          for (i=0; i < rx_pkt_count; i++) {
                    rte_prefetch0(rte_pktmbuf_mtod(rcv_pkt_bucket[i], void*));
                    struct ether_hdr *l2hdr;
                    l2hdr = rte_pktmbuf_mtod(rcv_pkt_bucket[i], struct ether_hdr*);

                    //=== BEGIN Your code

                    const struct ether_addr addr2set = {.addr_bytes={0x52,0x00,0x01,0x02,0x03,0x04}};
                    memcpy(&l2hdr->s_addr, &addr2set, sizeof(struct ether_addr));

                    //=== END Your code

                    int retrn = rte_eth_tx_burst(ETHDEV_ID, 0, rcv_pkt_bucket[i], 1);
                              //For better performance, bulk-send multiple packets.
                    if (retrn != 1)
                              RTE_LOG(INFO, USER1, "TX burst failed with error code %i.\n", retrn);
                    }

          }
}}
```

Retrieve pointers to the packet in bursts…

For every packet, do manipulations…

Mark the packets for sending…

DPDK Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Opportunities**
  - More performance
    - Avoiding kernel performance bottleneck by bypassing kernel packet I/O
    - Less jitter on dedicated cores
  - More, exclusive control over the network interface card
- **Challenges**
  - Security
    - No per-socket isolation
    - Userland code must be fully trusted on the kernel's machine (i.e. VM, bare-metal server)
    - Custom hardware C code bears the risk of buffer overflows
  - Not a network stack
    - Implement your own network stack, fitting the purpose of your network function
    - Yet hard to make use of TCP
- More information about DPDK under http://dpdk.org

DPDK in Action

DEMO: Writing, Compiling, and Running DPDK Applications

Improving NFV Performance: A Survey

NFV Performance Today

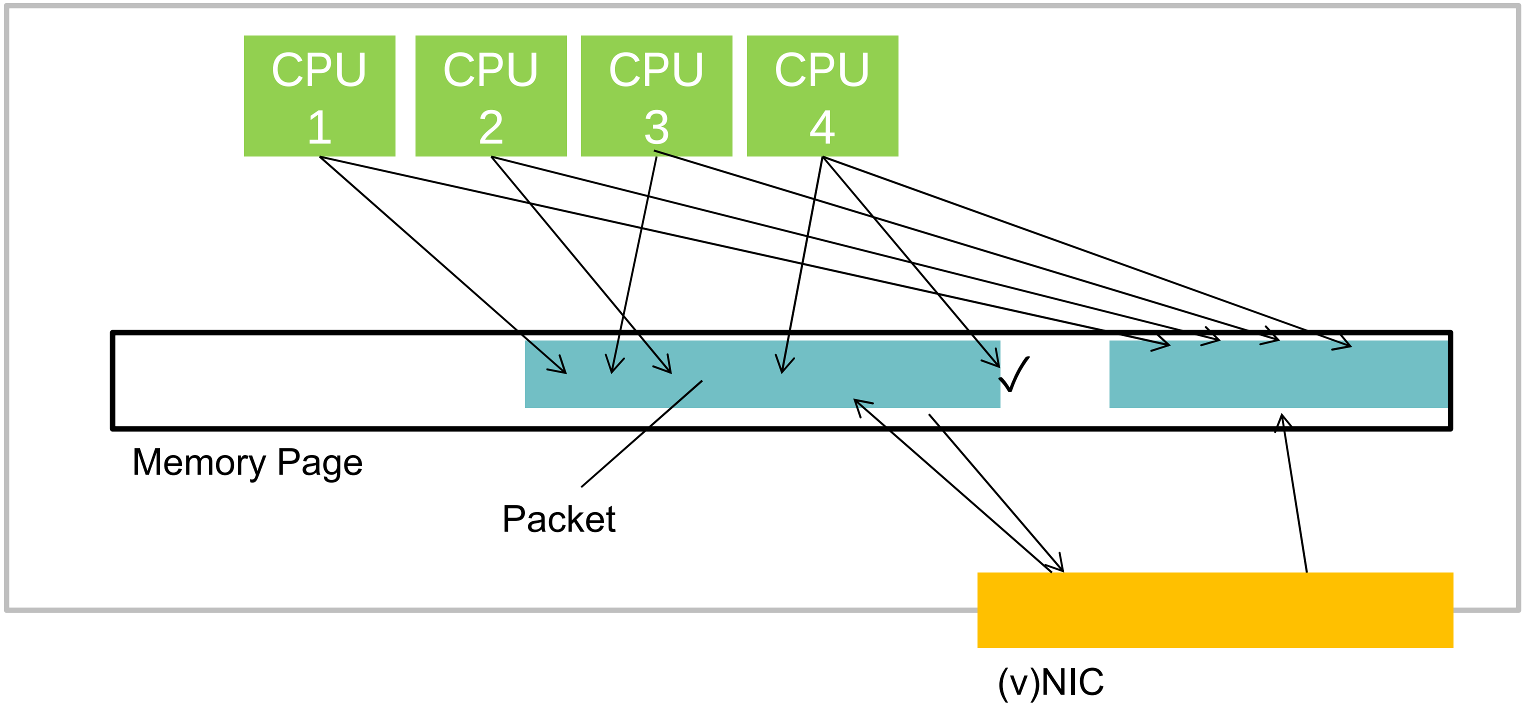
- More performance metrics!
  - Common: **delay, jitter, throughput (pps)**
  - Novel in NFV: **startup/migration time**

- Small-sized packets: Not suitable for >=10Gbps line-rate performance [S43]
  - Even when only forwarding packets!

- Approaches:

	Virtual Network Forwarding	HW-assisted Network Forwarding	Network Stack Offloading	VM Network I/O Optimization
Technologies	bridge [S7], Open vSwitch [S54, S53], MacVTap [S5], VALE/Netmap [S60]	SR-IOV, VMDq	DPDK [S19], ODP [S8], netmap [S60]	VirtIO [S6], PMDs, ELVIS, Interrupt Coalescing [S24]
Summary	**Zero-copy** forwarding primary target (e.g. IVSHMEM)	HW input queue management helps SW **parallelization**	Avoid **kernel** stacks. Use **SDKs** with HW offloading features	Use **poll-mode drivers (PMDs)** for best throughput. Delay may suffer.

Improved NFV Performance, Vision 1



- DMA and **zero-copy** forwarding (CPU-consuming)
- Parallelize packet operations (Improved delay)
- Poll-mode drivers (Improved throughput)

NFV Performance Research Challenges

- Many efforts to improve performance focused on **forwarding**
- Yet a challenge remains for VNFs...
  - ... which are **compute-intensive,** and
  - which operate on a packet's (entire) **payload**

- Present hardware uses **sequentially**-working processing cores
  - When receiving a packet, processor must "walk over" entire payload until packet can be sent out
  - Parallelization done with different packet queues, but not for tasks on same packet
  - Especially **delay** suffers from that

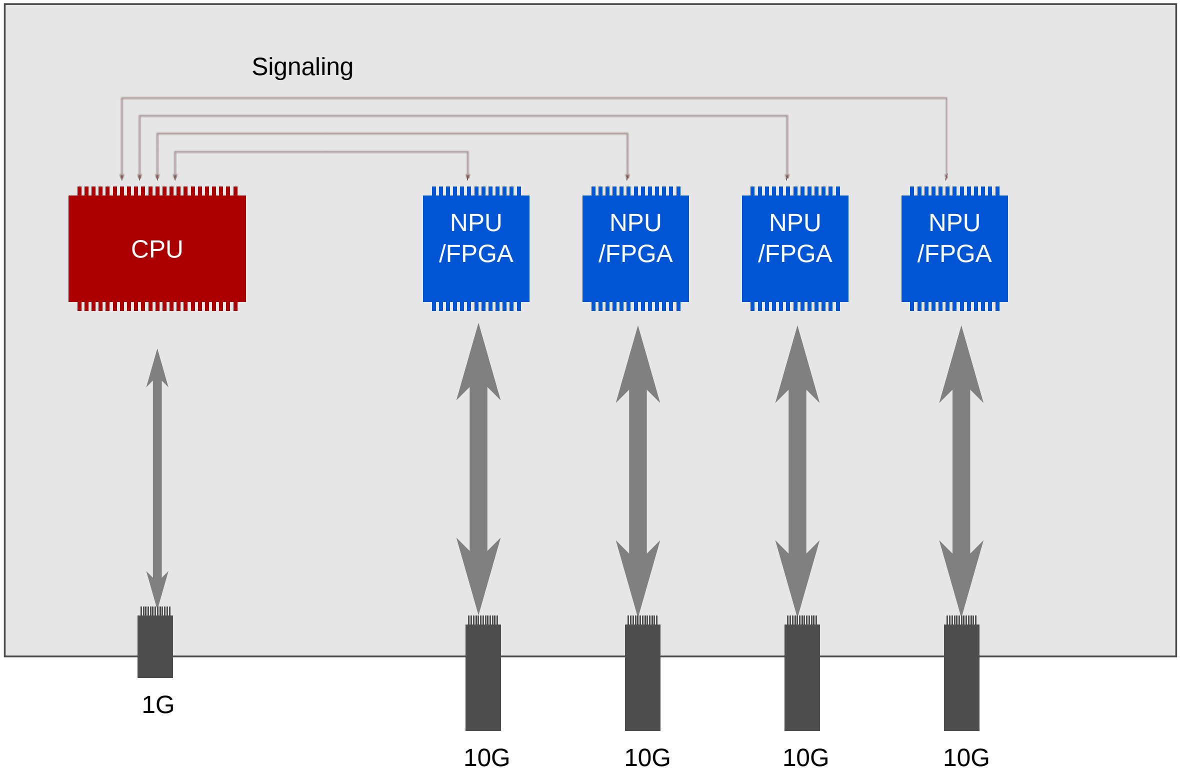
Improved NFV Performance, Vision 2

- Dataplane CPUs in Vision 1 often fulfill very simple-use-case tasks
- Another approach is to use **reconfigurable hardware** for these tasks
  - **Guaranteed** delay and line-rate throughput
  - **Parallelizable** to highest granularity
  - One clock step: thousands of operations!
  - Reprogramming allows for **IaaS-grade flexibility**
- **But**
  - Hard to implement
  - Proprietary toolchains

Acceleration Hardware (AH)

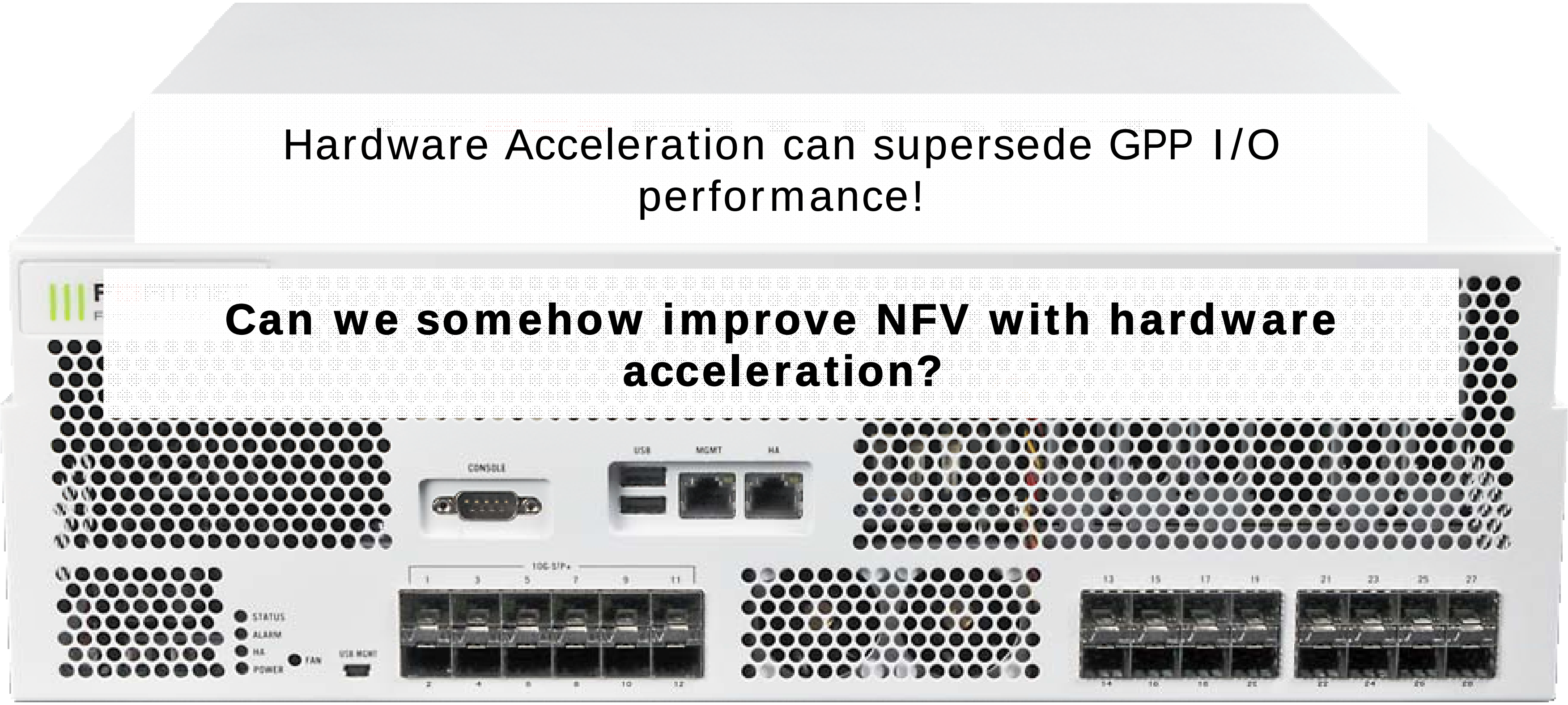
- Application-specific integrated circuits (ASICs)
  - Integrated circuits (IC) designed for a very special purpose
  - Highest performance
  - Not reprogrammable for different use case
- Field-Programmable Gate Arrays (FPGAs)
  - Re-programmable circuits
  - Lower performance than ASICs
  - Resource Sharing through **Partial Reconfiguration**
- Network Processors (NPUs)
  - Procedural processors optimized for packet processing
  - Performance improvement through *parallelization* and *pipelining*
- Graphics Processors (GPUs)
  - Single Instruction, Multiple Data (SIMD)
  - Not only for multimedia transcoding, also useful for several network tasks

Hardware Acceleration in Appliances



- Appliances offload tasks to acceleration hardware
- CPU only used for complex tasks with low bandwidth demands

Performance of Hardware Acceleration



Hardware Acceleration can supersede GPP I/O performance!

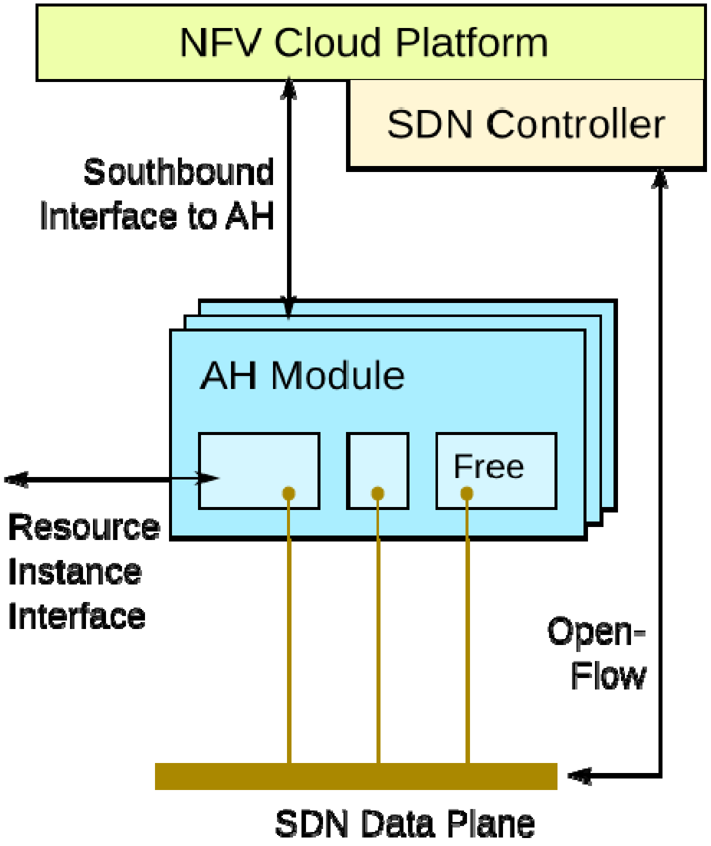
Can we somehow improve NFV with hardware acceleration?

GPP: General Purpose Processor

Source: http://forti-net.co.uk/media/wysiwyg/FG-3600C_firewall.png

Current Research: AH Modules [9]

- Consists of one or multiple AH processors
  - With dedicated network interfaces
- Tasks
  - Manage AH resources
  - Handle control plane communication
  - May have small co-processor for that
- Physical configurations
  - PCIe card on hypervisor
  - Standalone physical node
  - In an OpenFlow switch



Experimental Board: NetFPGA

References

[1] J. Martins et al. “ClickOS and the art of Network Function Virtualization”. USENIX NSDI 2014.

[2] G. Pongracz, et al., "Removing Roadblocks from SDN: OpenFlow Software Switch Performance on Intel DPDK“. EWSDN 2013.

[3] P. Emmerich, et al. "Performance Characteristics of Virtual Switching." CLOUDNET 2014.

[4] G. Pongrácz, et al. "Cheap silicon: a myth or reality? picking the right data plane hardware for software defined networking." ACM HotSDN 2013.

[5] C. Kachris, et al. "Network Function Virtualization based on FPGAs: A Framework for all-Programmable network devices." arXiv preprint arXiv:1406.0309 (2014).

[6] G. Brebner, “Softly defined networking,” ACM/IEEE Symposium on Architectures for Networking and Communications Systems 2012.

[7] X. Ge, et al. "OpenANFV: Accelerating Network Function Virtualization with a Consolidated Framework in OpenStack." ACM SIGCOMM 2014.

[8] S. Byma, et al. "FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack." IEEE FCCM 2014.

[9] L. Nobach, D. Hausheer: Open Elastic Provisioning of Hardware Acceleration in NFV Environments. SDNFlex 2015.

References (2)

[S5] Kernel Newbies: MacVTap. http://virt.kernelnewbies.org/MacVTap

[S6] KVM: VirtIO. http://www.linux-kvm.org/page/Virtio.

[S7] Linux Foundation: bridge. http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge.

[S8] OpenDataPlane - Project Website. http://www.opendataplane.org/.

[S19] I. Cerrato, M. Annarumma, and F. Risso. Supporting fine-grained network functions through Intel DPDK. In EWSDN, 2014.

[S24] Y. Dong, D. Xu, Y. Zhang, and G. Liao. Optimizing network I/O virtualization with efficient interrupt coalescing and virtual receive side scaling. In IEEE CLUSTER, 2011.

[S53] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby. Virtual switching in an era of advanced edges. In Workshop on Data Center–Converged and Virtual Ethernet Switching, 2010.

[S54] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending networking into the virtualization layer. In ACM HotNETs, 2009.

[S60] L. Rizzo and G. Lettieri. Vale, a switched ethernet for virtual machines. In ACM Conference on Emerging Networking Experiments and Technologies, 2012.