Open, Elastic Provisioning of Hardware Acceleration in NFV Environments

Leonhard Nobach, M. Sc.
lnobach@ps.tu-darmstadt.de

Prof. Dr. David Hausheer
hausheer@ps.tu-darmstadt.de

Netsys 2015
SDNFlex Workshop
Cottbus, Germany

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 318398.
Network Functions Virtualization (NFV)

Initiative of network operators
Specified by the European Telecommunications Standards Institute (ETSI) [2]

Idea: Put network functions onto commodity hardware

- Reduced procurement and maintenance costs
- Elasticity, Fast Time-to-Market, also through virtualization

Challenge: NFV Performance

- Virtualization overhead
  - Virtualization technology: Higher latency and reduced bandwidth
- Compete with hardware acceleration
  - Appliances use ASICs, NPUs and FPGAs
  - Often achieve line rate
Problem Motivation

NFV Performance - Solution Approaches

- **Scale-Out**
  - common mitigation, but requires more hardware, rack space and energy.
- **Reduction of Virtualization and I/O Overhead**
- **Hardware Acceleration**

---

ClickOS [3]
- Xen-based hypervisor and µkernels
- Results (small packets)
  - 45 µs delay (KVM: 65µs)
  - 7-8 Mpps (KVM/Xen out of the box: 0.8 Mpps) => 4096 Mbit/s

Data Plane Development Kit [4]
- Set of drivers and libraries to speed up network I/O processing
- Results very different
  - Intel*: 80 Mpps (W'load: bare throughput?)
  - Pongrácz et al. [5]: <11 Mpps (Workload: OpenFlow Software Switch)
  - Emmerich et al. [6]: <10 Mpps (Workload: DPDK Open vSwitch)
  - Results very dependent of network function!
Performance of Hardware Acceleration

Hardware Acceleration can supersede GPP I/O performance!

Can we somehow improve NFV with hardware acceleration?
Questions

❖ **Q1:** How much can **efficiency** (performance vs. costs) be increased by combining virtualized network functions (VNFs) with *network-attached* acceleration hardware (AH) to fulfill a common task?
   - **Elastic:** Seamless transition depending on current performance needs.
   - Network attachment: Requirement for “pooling” AH resources.

❖ **Q2:** In the scenario described in Q2, is it possible to support all state-of-the-art *network-attached* hardware acceleration technologies with an **open** framework?
   - Competition of hardware vendors
   - Opportunity of **two-sided market**: AH vendors and network function developers

**GPP:** General Purpose Processor
Background

**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)
Background

AH and Use Case Complexity [7]

- Performance of NPU decreases with use case complexity (Pongracz et al.)
- We conclude: Work offloaded to AH should have simple use case
  - Opportunity for hybrid GPP/AH architecture

AH: Acceleration Hardware, GPP: General Purpose Processor
## Comparison of Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA-only</td>
<td>FPGA-only</td>
<td>FPGA-only</td>
<td>FPGA-only</td>
<td>FPGA-only</td>
<td>All (FPGA, NPU, GPU), if network-attached</td>
</tr>
<tr>
<td>Hybrid GPP/AH</td>
<td>AH-only</td>
<td>No (not specified)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Full Programmability</td>
<td>Yes</td>
<td>Yes (high-level language)</td>
<td>No, fixed set</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Location Independence</td>
<td>On certain data plane devices</td>
<td>On certain data plane devices</td>
<td>AH modules on same computing node</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AH Pooling</td>
<td>(not specified)</td>
<td>(not specified)</td>
<td>AH pooling on same computing node</td>
<td>AH pooling in datacenter</td>
<td>AH pooling in datacenter</td>
</tr>
<tr>
<td>Elasticity</td>
<td>(not specified)</td>
<td>(not specified)</td>
<td>(not specified)</td>
<td>Manual provisioning</td>
<td>On-demand</td>
</tr>
</tbody>
</table>

**AH**: Acceleration Hardware, **GPP**: General Purpose Processor
Elastic Provisioning of Hardware Acceleration Resources in NFV Environments

APPROACH
Approach

Use Case Separation

- **Low utilization: Classical NFV**
  - Whole traffic handled by network function on GPP
- **High utilization: Traffic with simple use case directed to AH resource**
- **OpenFlow/OF-PI switches direct traffic according to VNF-specific rules**

**GPP**: General Purpose Processor, **AH**: Acceleration Hardware
Network Functions: UC Separation Examples

- **Point-to-Point (PPP) Access Concentrator** *(RFC 2516)*
  - GPP: Discovery/Authentication stage
  - AH: Session stage

- **IPSec Endpoint**
  - GPP: Authentication, Key Exchange
  - AH: Payload Encryption, Signing

- **Session Border Controller (SBC)** *(RFC 5853)*

- **CDN Cache**

**GPP:** General Purpose Processor (e.g. x86, ARM), **AH:** Acceleration Hardware (e.g. FPGA, NPU, GPU)
Approach

Use Case Separation and Elasticity

GPP: General Purpose Processor, AH: Acceleration Hardware

Seamless Transition
Elastic Provisioning of Hardware Acceleration Resources in NFV Environments

SYSTEM ARCHITECTURE
Network Function – Software Package

- Network function is delivered as package
- Main software (GPP)
  - Code for complex UC
  - Code for simple UC (Used under low utilization)
  - Instructions for SDN traffic splitting
- Offloading program(s)
  - Code for simple UC
  - High-level language or platform-dependent

**Main Program (C/Python/…)**

- Offloading Program (HDL for FPGA)
- Offloading Program (NPU Code)

**GPP**: General Purpose Processor, **HDL**: Hardware Description Language
Overview

- Builds upon GPP NFV platform
  - Candidate: OpenStack
  - ClickOS/DPDK can be used

```
<table>
<thead>
<tr>
<th>NFV Cloud Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDN Controller</td>
</tr>
</tbody>
</table>

VNF

- Southbound Interface to VNF

AH Module

- Southbound Interface to AH

Resource Instance Interface

GPP Traffic

AH Traffic

SDN Data Plane

OpenFlow
```
AH Modules

- Consist 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
**AH Southbound Interface**
- AH modules register at SDN controller with resources, capabilities and network links
- Controller reserves AH resources
- Initiates: VNF-AH (over RII)

**VNF Southbound Interface**
- VNF acquires resources (with compat. description)
- SDN controller provides VNF-AH (over RII)
- Control redirection to AH for specific sessions

**AH Resource Instance Interface (RII)**
- Used to program the AH resource
- Used to configure AH resource (e.g. sessions in CAM table over “DMA-over-Ethernet”)

---

**Terms:**
- **VNF:** Virtual Network Function
- **AH:** Acceleration Hardware
- **GPP:** General Purpose Processor
- **HDL:** Hardware Description Language
Elastic Provisioning of Hardware Acceleration Resources in NFV Environments

CONCLUSION, FUTURE WORK
Novel approach for the provisioning of hardware acceleration

- **Elastic:**
  - as VNFs can acquire as much hardware acceleration as currently needed for their task.

- **Hybrid:**
  - Workload of VNF is partially carried out by the main processor and acceleration hardware

- **Open:**
  - every network function can provide its own acceleration description for a variety of different hardware, competition of hardware vendors
Conclusion, Future Work

Current and Next Steps

- Investigation: Use case separation of different network functions and its implications
  - Seamless state transfer GPP <-> AH, even whole TCP sessions
  - Network delay between GPP and AH may lead to protocol inconsistency
  - OpenFlow traffic splitting

- Description languages for certain groups of hardware (e.g. FPGA -> VHDL, NPU -> PacketC?)
  - Selection, modification or development from scratch

- Survey and comparison of commonalities of accelerated hardware

- Prototype implementation in a testbed, performance and elasticity measurement
  - Candidate: OpenStack + Ryu
Conclusion, Future Work

First AH Module: NetFPGA SUME
Thank you for your attention!

P2P Systems Engineering Lab - PS

Leonhard Nobach, M.Sc.
Department of Electrical Engineering and Information Technology

Rundeturmstr. 12
64283 Darmstadt
Germany

Phone +49 (0) 6151/164959
lnobach@ps.tu-darmstadt.de
www.ps.tu-darmstadt.de
References


References (2)
