Dynamic accelerator provisioning for SSH tunnels in NFV environments

Gourav Prateek Sharma, Wouter Tavernier, Didier Colle, Mario Pickavet

Ghent University - IMEC, IDLab, Department of Information Technology

Email: gouravprateek.sharma@ugent.be

***Abstract*. In this demonstration, we present dynamic allocation of accelerator resources to SSH tunnels in an NFV environment. In order to accelerate a VNF, its compute-intensive operations are offloaded to hardware cores running on an FPGA. The CPU utilization information of VNFs is continuously processed by a service management component to dynamically decide the suitable target to run VNF's crypto-operations. We also demonstrate switching between the non-accelerated and hardware-accelerated SSH-tunnels triggered by a change in the nature of the data traffic flowing through the tunnel and indicate throughput gains obtainable in dynamically switching contexts.**

Keywords—SSH, tunneling, NFV, FPGA, acceleration

I. INTRODUCTION

Traditionally, network services are deployed using specialized and proprietary hardware boxes known as middleboxes. Network function virtualization (NFV) aims to decouple the functionality of these middleboxes from the underlying hardware so that standard IT virtualization technologies can be exploited to execute network functions on general-purpose x86 or ARM servers. NFV has enabled faster deployment of new network services along with a reduction in expenditures. Despite all the benefits that NFV offers, it still faces challenges towards its widespread acceptance. One of the major challenges is to achieve the same virtual network function (VNF) performance as offered by its hardware counterpart [1]. To overcome this challenge, the use of hardware accelerators along with general-purpose processors has been proposed [2, 3]. As NFV aims to decouple network functionality from dedicated middleboxes, reconfigurable hardware platforms like FPGAs acting as hardware accelerators are gaining particular attention. FPGAs offer the best of both worlds, i.e., programmability of general purpose processors and performance of dedicated hardware boxes. Therefore, compute-intensive portions of a network function could be offloaded to accelerators running on an FPGA.

SSH tunneling service is chosen to show the proof of concept of hardware acceleration for a virtual customer premise equipment (vCPE), as an NFV use case. The bandwidth of an SSH tunnel is limited by the speed with which a CPU can perform cryptographic operations on data packets. These operations include bit- and byte-level manipulations like XOR, byte substitutions and shifting. By utilizing parallelism available on FPGAs, these operations can be performed much faster on a hardware platform, in comparison to software executing sequential instructions on a CPU. Moreover, overall operating costs can be reduced as FPGAs are more efficient than CPU in terms of number of operations/Joule of energy spent. However, effectively integrating accelerator resources and then optimally allocating them in the NFV environment is still an challenge. As the resources present on a CPE are limited, a strategy is required to grant accelerator resources to network functions based on their real-time resource utilization.

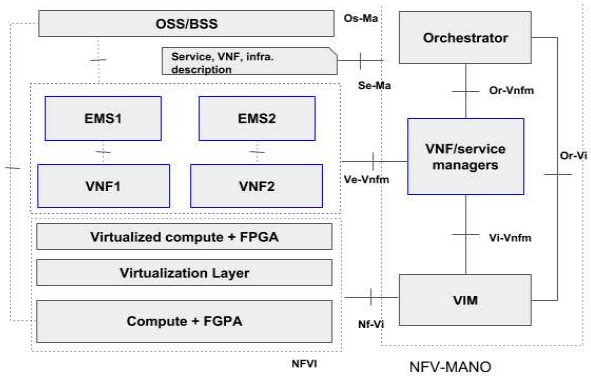

Fig. 1 ETSI NFV reference architecture framework.

Fig. 1 shows the reference NFV architecture proposed by the ETSI. The key components relevant for our demonstration are highlighted in blue. The VNF/specific service manager (VNFM or SSM) is responsible for the life-cycle management, i.e. starting/stopping, scaling and configuring, of one or more VNFs. The element management system corresponding to each VNF and VNF/service managers work together to configure the application specific parameters of VNFs during its lifetime. In our demonstration, we show how accelerator resources could be dynamically activated for VNFs with the help of EMSs and VNF/service managers.

II. SYSTEM IMPLEMENTATION AND PROCESSES

It is often required to extend network services present inside a private network of an enterprise to a host located in the public network. SSH-tunneling is a secure way of communication between two machines over an untrusted network. An SSH tunnel works by forwarding packets received at a configured port on a client machine to a specific port on the destination server machine (local port forwarding). An SSH-client listening

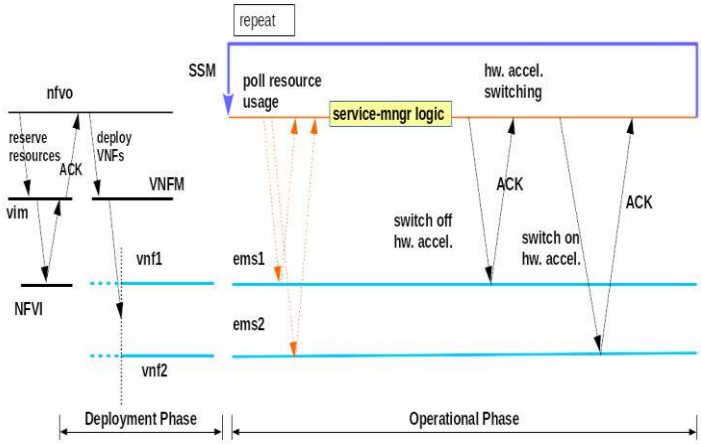

Fig. 2 Sequence diagram showing processes involved in deployment and management of hardware accelerated VNFs

on a particular port creates a secured encrypted channel to the SSH-server which in turn forwards the traffic to a port on the destination machine present in the private network.

In our implementation, we choose Dropbear SSH-client[1] as a network function which along-with the required configuration are packaged as docker images. An identity file (RSA public key) required by the SSH-client to authenticate itself to the SSH server is also present in the docker image.

A. Service Management Processes

The processes involved in the deployment and management of hardware accelerated VNFs are explained in Fig. 2. First, component VNFs of the required service are deployed by NFV orchestrator by delegating resource reservation to the Virtual Insfrastructure Manangement (VIM) system and then VNF instantiation task to VNFMs. The orchestrator should ask VIM to also reserve accelerator resources for specific VNFs, e.g., hardware accelerator cores for SSH-client VNFs. Next, each VNF is polled by the SSM to fetch its resource utilization information. Based on this information, the SSM logic determines the suitable candidate VNF for allocating the accelerator. The SSM logic involves a hysteresis decision loop with pre-defined upper and lower thresholds for CPU utilization. Before granting accelerator to a VNF, currently accelerated VNF is signaled to release the accelerator resource.

B. Hardware Accelerator System Design

The original Dropbear implementation utilizes ciphers and hashes provided by libtomcrypt (cryptographic library), to en/decrypt and hash data packets. In order to accelerate these cryptographic operations in dropbear, FPGA based accelerators are utilized. Dropbear code is patched with accelerator functions for communication with AES-128 and SHA-256 accelerator cores. These accelerator cores are based on an open-source Verilog implementation [4]. The system architecture for accelerating en/decrypting and hashing using external hardware cores is shown in Fig. 3. For en/decryption, the hardware core is first initialized by writing keys and initialization vectors into its memory-mapped registers. Similarly, for hash initialization, the current hash state is written to SHA-256 core's registers. The initialization is followed by the transfer of plaintext from the RAM memory to the accelerator core where ciphertext or hash is calculated. The progress of the core is monitored continuously by checking the "progress pointer" of the corresponding core. After the processing of the plaintext is complete, ciphertext or hash is transferred back into the memory. All data transfer tasks between the memory and accelerator cores are managed by the direct memory access (DMA) controller present on the ZYNQ processing system (PS). The hardware design for the AES-128 and SHA-256 accelerators was developed and implemented in the Vivado environment. A kernel module is used to manage DMA transfers from the userspace buffers in Dropbear to the respective cores using zero-copy mechanisms[2].

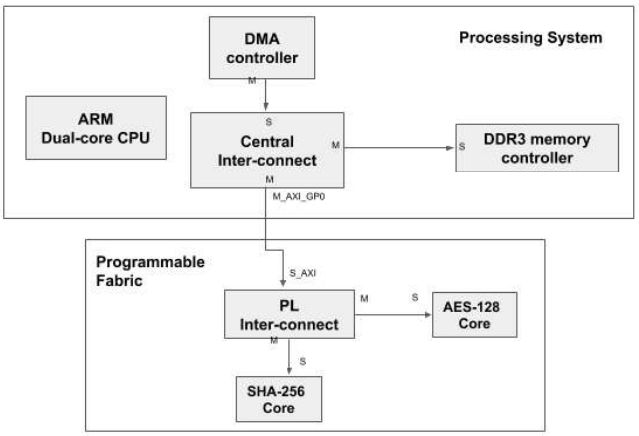

Fig. 3 System design for hardware acceleration of AES128 and SHA256 .

III. Demonstration Description

The demonstration is a representation of a scenario wherein a home-user would like to establish a secure channel to a machine its office or a data-center. The demonstration setup consists of a PC acting and a PYNQ board. The docker engine running on the PYNQ board forms NFVI, VIM is implemented by docker tools, and VNFs used in the demonstration are docker containers, each running two applications:

1. **Dropbear SSH client (dbclient)**
2. **Element management (em)**

In our setup, the EM is responsible for two tasks: (1) replying to SSM's inquiry about CPU usage and (2) signaling dbclient on behalf of SSM to use/release accelerator resources. The demonstration consists of two parts-- first part outlines the performance comparison between original, and hardware accelerated SSH-tunnels and the second part demonstrates

[1] https://github.com/mkj/dropbear

[2] https://github.com/jeremytrimble/ezdma

mechanism for accelerator allocation to VNFs based on their CPU utilization. The setup for the demonstration is shown in the Fig. 4.

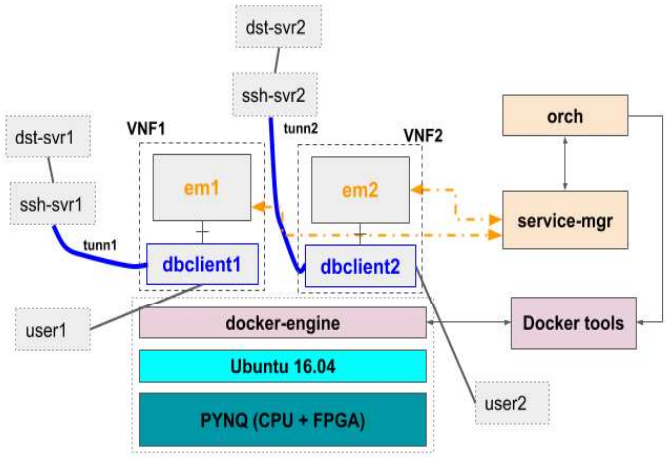

Fig. 4 Demonstration setup for dynamic acceleration of SSH tunnels.

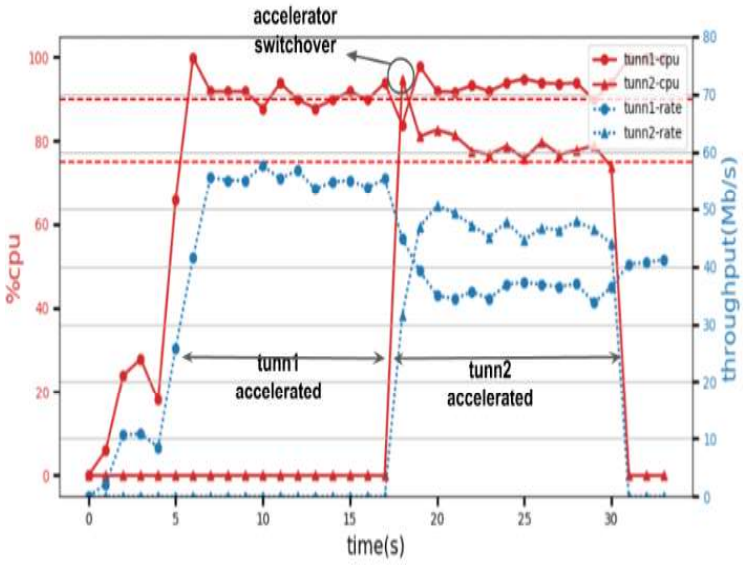

Fig. 5 Time variation of CPU usage and throughput values for two SSH tunnels.

A. Performance comparison

In the first part, a SSH tunnel is established between the PYNQ board and the PC by instantiating VNF1 only. As a result of port forwarding, traffic sent to port X1 on PYNQ is forwarded to port Y1 on the laptop, where destination server (iperf server) is listening. An user, simulated by an iperf client, starts sending traffic to the port X1 on PYNQ board. The peak throughput of the SSH tunnel is displayed on the screen and top displays CPU usage of the running SSH-tunnel. Then a curl request is manually sent to em1 to use the hardware accelerator instead of CPU. We will observe an improvement of about 40% in the traffic flowing through the tunnel. Moreover, the CPU utilization by SSH tunnel running on the PYNQ board also drops from 99% to 86%.

B. Dynamic accelerator allocation using SSM

In this case, two VNFs with dbclient1 and dbclient2 are deployed on the PYNQ board resulting in two separate SSH tunnels, i.e., tunn1 and tunn2 respectively. The second tunnel acts as a backup secure connection to the destination machine. CPU usage of both tunnels is continuously monitored and fed to the SSM running on the PC. Initially, a user creates a traffic with traffic rate of 10 Mbps is started on tunn1 resulting in the CPU utilization well below the upper threshold level of 90%. After a few seconds, traffic on tunn1 is increased such that it is completely saturated and its CPU usage reaches 99%. As a consequence, the SSM, via em1, triggers the dbclient1 to use AES-128 and SHA-256 hardware cores. CPU is therefore relieved from performing compute-intensive operations and its utilization drops to about 88%. For demonstrating accelerator switching, user2 starts sending traffic on the tunn2. This results in SSM taking accelerator from tunn1 and allocating it to the tunn2. It can be clearly observed that the peak throughput of the tunn2 tunnel is increased by about 40%, accompanied by a reduction in CPU usage. The variation in traffic rate (blue) and CPU usages (red) of two tunnels is shown in Fig. 5. The peaks in CPU usage of tunn1 and tunn2 at t = 6s and 18s, respectively, are processed by the SSM to produce triggers for the two VNFs to use the accelerator.

IV. CONCLUSION

The challenge of effectively utilizing hardware accelerators in NFV environments can be addressed by the co-operation between EMs and the SSM components. SSM is responsible for the provisioning of accelerator resources to network functions by taking into account their real-time resource utilization. We demonstrated this by dynamically activating hardware accelerator cores for two SSH-tunnels. AES-128 and SHA-256 operations of SSH-tunnels were offloaded to hardware accelerator cores running on an FPGA resulting in improvement of their peak throughput and conservation of CPU cycles.

V. ACKNOWLEDGMENT

This work has been performed within the framework of the NGPaaS project, which is funded by the European Commission through the Horizon 2020 and 5G-PPP programs.

VI. REFERENCES

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236-262, Sept. 2015.

[2] C. Kachris, G. Ch. Sirakoulis, D. Soudris, "Network Function Virtualization based on FPGAs:A Framework for all-Programmable network devices", in *arXiv*, June 2014.

[3] X.Ge, Y. Liu, D.H.C. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, X. Hu, "OpenANFV: Accelerating network function virtualization with a consolidated framework in OpenStack", in *2014 ACM Conference on Special Interest Group on Data Communication,* Aug. 2014.

[4] J. Strombergson. Verilog implementation of the symmetric block cipher AES and SHA. [Online]. Available: https://github.com/secworks/