The Journal of Supercomputing (2022) 78:1565–1582
https://doi.org/10.1007/s11227-021-03907-0

A measurement study of offloading virtual network functions to the edge

S. R. Chaudhry[1] · P. Liu[3] · X. Wang[3] · V. Cahill[2] · M. Collier[3]

Accepted: 21 May 2021 / Published online: 14 June 2021

Abstract
The deployment of virtual network functions (VNFs) at edge servers potentially impairs the performance of latency-sensitive applications due to their computational cost. This work considers a new approach to addressing this problem that provides line rate acceleration of VNFs by employing field-programmable gate array (FPGA) equipped edge servers. This approach has been validated by practical use cases with both TCP and UDP as underlying protocol on a physical testbed environment. We examine the performance implications of executing a security VNF at an FPGA-equipped edge server. We experimentally demonstrate reduced VNF execution latency and energy consumption for a real-time video streaming application in comparison with a software-only baseline. In particular, the results show that the approach lowers VNF execution latency and power consumption at the edge by up to 44% and 76%, respectively, in our experiments while satisfying time constraints and maintaining confidentiality with high scalability. We also highlight the potential research challenges to make this approach viable in practice.

Keywords Edge computing · Virtual network function (VNF) · Hardware acceleration · Data encryption · Task offloading

1 Introduction

With the exponential increase in the number of devices connected to the Internet of Things (IoT), the fog and edge [1] computing paradigms are gaining momentum, for example, in 5G technology specifications [2, 3]. Both bring cloud-like applications closer to end users and 'things' at the edge of cellular networks, enabling

✉ S. R. Chaudhry
saqib.chaudhry@cit.ie

[1] Department of Computer Science, Munster Technological University, Cork, Ireland

[2] School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland

[3] School of Electronic Engineering, Dublin City University, Dublin, Ireland

new functions (services) with challenging performance requirements for latency and security that cannot be met by current cloud solutions running on remote data-center sites [4]. Radically new applications, such as the Tactile Internet, real-time autonomic control, augmented reality, Industry 4.0, e-Health, smart cities/grids, or autonomous vehicle platooning, are key representative use cases to be supported by these emerging IoT architectures [5].

Multiple paradigms including cloudlets, fog and edge Computing have been proposed to address challenging requirements of latency and security in different application contexts, but all share the common idea of reducing the dependency on the remote cloud. These computing paradigms have opened many new frontiers for versatile and uninterrupted services to be provided as IoT applications. Edge computing empowers tiny IoT devices with significant additional computational capabilities through computation offloading supporting acceleration of critical tasks. Similarly, IoT expands edge computing services to all types of smart objects ranging from sensors and actuators to smart and (semi) autonomous vehicles. One of the key benefits of merging IoT and edge computing is lower latency due to the reduced physical and virtual communication distance in contrast to using remote clouds. However, the computational resources available at the edge for compute-intensive tasks such as data security/privacy for real-time applications are nevertheless limited although offering the advantages of short access distance, flexible geographical distribution and relatively richer computational resources than IoT devices.

Smart things equipped with tiny sensors (camera) are different from other things due to the critical and sensitive nature of the information and data-rate requirements for the video transfer. Thus, the interconnection of things with cameras is types of IoT devices used in various applications ranging medical, surveillance, smart mobility, etc. Lack of security implementation on such interconnected IoT devices makes the surveillance applications unsecure, because everything is Internet accessible in the IoT world. Information security of such Internet enabled devices is vital to build trust and consumer confidence in their use. Considering, consumers of these devices believe that their critical and personal information are not secure and can be misused or harmful, then they will be reluctant to use IoT devices. Despite the fact stated, it is also crucial that enabling information security does not degrade the performance of the time-sensitive IoT systems. Information security in the IoT systems is not only an important, but also challenging aspect, ranging from the technological to the psychological issues, such as privacy and trust. Knowing this can classify the IoT security threats into two categories such as physical and non-physical attacks. The physical attacks are threats generated by physical harming the things that are publicly deployed in the common accessible areas and hard to be supervised constantly. The non-physical attacks are threats to the things when they start communications. These attacks range from eavesdropping, man-in-the-middle attack, routing information compromise, denial of services, etc. Securing real-time video information transmission from eavesdropping and man-in-the-middle attack is addressed in this paper.

In recent decade, micro-controller integrated devices, referred as “Things” are transformed from very simple and low-power devices to computational complex devices requiring power-hungry sensors, bandwidth demanding communication and contain

various hardware platform and architecture (e.g., field-programmable gate arrays). The key advantage of FPGA platforms over application-specific integrated circuit (ASIC) is programming and design flexibility, which makes an FPGA a recognised and popular hardware for IoT devices [6]. This enables the FPGA an easy to interface platform by integrating temperature, pressure, position, acceleration, ADC and DAC converters, since these interfaces are required by new and smart "Things". Another major reason for FPGA in things is long life spans of IoT devices. Vendors may stop developing and rolling out software patches for a prototype/product if it reaches obsolescence. This motivates why security service should be implemented with re-programmable/configurable hardware solutions (such as FPGAs) rather software patches. To facilitate such deployment, the prime contributions of this work can be summarized as:

- We combine the native advantages of edge computing and FPGA acceleration to effectively support encrypting video streams, while benefiting time-sensitive and bandwidth-hungry IoT applications for TCP and UDP traffic.
- We design and implement a proof-of-concept FPGA-equipped edge server. The testbed demonstrates that the approach can reduce the latency and power consumption of VNF execution by up to 44% and 76%, respectively, against software-only edge/cloud baselines.
- We identify the open challenges posed by this merger of these technologies.

The rest of this paper is organised as follows. Section 2 presents requirements for number of IoT use cases and significance of computational offloading and hardware acceleration for these. Section 3 highlights the benefits of VNFs deployment at the edge and prime contributions of the work. Section 4 discusses the experimental testbed, physical components, and hardware specifications for the proposed system. Section 5 covers experimental scenarios and metrics with performance evaluations and analysis. Section 6 discusses the related work. Section 7 presents the conclusion and potential research challenges realised in this study.

2 Motivating offloading and acceleration at the edge for IoT use cases

Computation offloading for IoT applications transfers resource-intensive tasks to an external platform, such as a cluster, grid, cloud, fog, edge server, or user device, including a processor, graphics processing unit (GPU) or field-programmable gate array (FPGA). Acceleration and offloading are necessary due to the current hardware limitations of IoT devices and servers. The offloading decision can be based on network specifications (4G/5G/WiFi/WiMAX, available bandwidth), application requirements, server specifications (CPU, storage/memory) or user preferences (data confidentiality, reliability, experience).

Nowadays, it is common for applications such as commercial speech-based intelligent personal assistants, such as Apple's Siri, which employs a deep neural network as its speech recognition model, to employ computation offloading to the cloud [7]. However, as the network conditions between user devices and remote clouds

are unreliable, offloading today's compute-intensive applications may lead to high latency and result in poor user experience.

2.1 Why offload to the edge?

By processing data close to consumers/producers, edge computing is proving to be a promising solution [3] for addressing the limitations of the cloud in supporting delay-sensitive and context-aware services. Addressing these limitations are crucial for real-time applications, which not only rely on compute-intensive tasks and algorithms such as data aggregation/fusion, but also require timely responses. Recent work [4, 7, 8] has experimentally quantified the benefits of offloading to the edge. For instance, by placing VM-based cloudlets at the edge to accelerate the computational engine, [8] achieved a $4.9\times$ lower response time when compared with cloud-based offloading, enhancing the user experience and also saving mobile device power [9].

2.2 Why use FPGAs for acceleration?

The use of FPGA-based accelerators is well established in the application context. For example, FPGA-based acceleration for convolutional neural networks (CNN) can achieve $2x-2.5\times$ improvement in processing time for image classification over parallel GPUs [10]. GPUs have many cores (simpler than a CPU core) because they do similar processing at the same time. For instance, rendering an image is a matter of coloring lots of pixels and the process of doing that involves the same type of data being processed in parallel. GPU cores are specifically designed to do that type of processing many times per second. In another example, Sirius, an intelligent personal assistant [6], uses FPGAs to accelerate visual/speech processing workloads running in datacentres and achieves a query latency reduction of $16x$. In [10–12], researchers compare hardware platforms to characterize their performance acceleration for CNNs and binarized neural networks (BNN) to show that FPGAs are an appealing solution. Moreover, FPGAs have already been employed for cloud computation acceleration in datacentres by Microsoft [11].

2.3 Why network functions on programmable hardware?

The use of virtual network functions (VNFs) has emerged as a transformative technology that will allow service providers to move to a truly virtualized network infrastructure. VNFs will replace proprietary networking hardware with software functions deployed on open hardware platforms such as GPU, Smart NICs or FPGAs allowing increased agility, value and performance. VNFs such as encryption algorithms are computationally intense and expensive due to their complexity and iteration requirements, essentially consuming the energy, processing and computational abilities of devices and servers for time-critical applications, but making them prime candidates for offloading. The use of programmable hardware facilitates on-chip

logic to be reconfigured with dedicated pipelines and parallelism for performance enhancement (i.e., low latency and high throughput).

2.4 Use cases and requirements

The automotive and transportation sectors in smart and connected cities are experiencing a transformation due to innovative technologies enabled by increased connectivity [13, 14], but the communication requirements such as latency, reliability and data confidentiality are being failed by current infrastructure support [14]. The application-specific requirements for location-based services, traffic information, personal multimedia sharing and forwarding demand a rapid and secure service at the edge. In cloud computing environments, all such diverse data are uploaded to the cloud for analysis and learning for appropriate measures and information retrieval. Considering a range of transportation modes and their applications in a smart city, edge solutions are expected to have low cost, low energy consumption, high quality of experience, increased flexibility, high security and privacy and multivendor interoperability.

Numerous IoT devices will transform our homes/buildings in smart and intelligent pervasive surroundings, ranging from kitchen appliances to ambient light controllers. A key factor is their enhanced inter-working to cooperate with their users and cloud-hosted applications. One challenging aspect of this connectivity relates to the new potential security threats that attackers can leverage to conduct their malicious activities. For example, in October 2016, exploiting firmware security flaws, cybercriminals launched a distributed denial of service (DDoS) attack using many smart-home IoT devices, against a Domain Name System (DNS) provider ISP Dyn, thus disrupting major Internet platforms and services to large swathes of users in the EU and North America [15].

The fourth industrial revolution is the digitalization of industrial production processes [16]. To bring this to realization requires integrating industrial IoT, cyber-physical systems (CPSs) and computing technologies [17]. Industry 4.0 involves a hyper-connected system that includes the use of AI and robotics-based automation in industries to provide real-time information about operational behavior for quality assurance and system maintenance. The increased connectivity of industrial systems is a key for next-generation industrial standardization. The increasing openness, in Industry 4.0, will inevitably lead to significant security concerns. Indeed, the large number and the disparity of industrial equipment will lead to a myriad of security vulnerabilities exploitable by cyber attackers. Industrial information is of utmost importance for the competitiveness of a company. These concerns will greatly undermine the overall digitalization of industries.

Based on the stated use cases, requirements and benefits, this work studied the viability (i.e., its suitability for IoT applications) of using the security VNF deployed on FPGAs to support edge computing.

3 Offloading VNFs at the edge

This work is inspired by the potential cumulative advantages of VNFs, edge computing/offloading and use of FPGAs acceleration. The proposed approach aims to support edge computing to accelerate VNFs by offloading to networked FPGAs. In

a traditional edge computing architecture, the edge server processes the data before forwarding it to other edge servers or remote cloud sites. In our offloading approach, the incoming data at the cellular base station (BS), i.e., 4G-radio access network (RAN), are processed locally, at line rate, by an FPGA-equipped edge server executing a VNF such as encryption before forwarding it to the destination in order to negate potential security threats/risks posed by the public Internet.

In the IoT context, a natural question that arises is *whether the implementation of VNFs using FPGAs can allow network functions executed at burdened edge servers to satisfy application requirements?* To address this question, we conducted an experimental study of an encryption VNF to be deployed at an edge server equipped with a networked FPGA to compare performance against a traditional edge server using a physical testbed. To the best of our knowledge, this work is the first attempt to implement a VNF in a FPGA for edge computing to preserve data security and privacy while maintaining application latency requirements.

The application of FPGAs at the edge to support VNFs is a new topic and the state-of-the-art is still very limited. However, there has been some work on accelerating complex but non-VNF implementations using FPGAs [8–12], which also lacks assessment in an Edge environment. Researchers in [18] offload the processing of compute-intensive neural network applications to FPGAs. Their work established a comprehensive understanding of offloading possibilities but does not offer networked FPGAs for VNF implementation at the edge and lacks assessment of the implications and performance of any real-time data processing (for example, as desired in encrypting video streams at line rate). In our implementation, FPGAs do not simply offload/onboard VNFs but also support parallel data processing by pipelining the design in contrast to traditional proprietary hardware solutions. This allows different phases of a network function to work in parallel for data processing at line rate. These characteristics are significant in a reliable and scalable transition from cloud to edge as considered below.

Compared to VNF, edge computing addresses the need to deploy third-party applications. Edge computing allows application deployment inside each environment to be managed by the environment operators. Nesting of virtualization technologies allow edge operators to support multiple environments by allocating a VNF to each third-party application environment owner. Each third-party application environment owner can further allocate resources assigned to VNF to the other applications and services.

4 Video streaming testbed

Our study addresses video streaming as a representative use case in which this approach to deploying network functions (in this case, data security) is imminent and offers significant advantages.

According to Cisco VNI [19], camera-based applications will be the largest market for next-generation IoT solutions worldwide exemplified by the widespread use of video sensor networks. These systems are already becoming deeply interwoven into our lives given the significant focus on making homes, cities, electric grids,

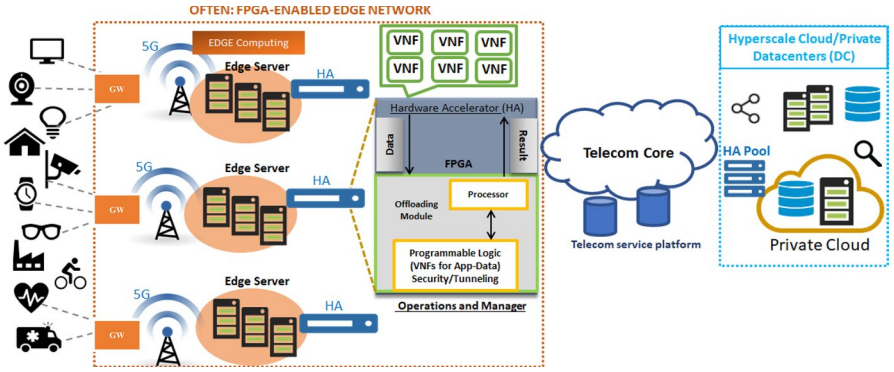

Fig. 1 High-level system diagram of proposed edge-cloud environment

factories, and health services among others, safer, more efficient, and easier to monitor and manage [14]. Considering the costly implementation and complexity of traditional camera deployments, which are dependent on cable and fiber connections to support these services at scale, wireless low-powered cameras significantly simplify the implementation and management complexity of such deployments in dense and urban communities. Additionally, newer form factors that use body-attached, vehicle-mounted or drone-mounted cameras can only be connected wirelessly. Such devices will often communicate with a local gateway (GW) which in turns offers a multiplexed wireless connection toward the wider internet. The major question around wireless as the connectivity medium of choice for video networks until now has been whether it can support the required bandwidth and latency requirements for real-time, high-definition camera feeds without compromising security and privacy requirements. Protection of video data is always critical due to the privacy concerns of citizens and such data should only be accessed by authorized entities in video surveillance applications.

While bandwidth limitations may be solved by the expected widespread deployment of 5G networks, ensuring the security and privacy of the video contents are particularly challenging, especially for resource-constrained GWs due to finite energy supply and low-computing power. These factors are typically at odds with most existing security protocols and schemes proposed for the IoT because of the intensive computational nature of the cryptographic algorithms involved. In addition, leaving IoT data unencrypted can exploit vulnerabilities in IoT devices.

Offloading can provide assured protection on demand for personal data (audio, video and identification) without putting further load on edge servers. Moreover, FPGAs allow hardware rules to be updated easily when compared to proprietary hardware. The ability to deploy custom policies/rules at the edge server using a FPGA can also be a performance advantage over software alternatives. Thus, our work also provides an experimental demonstration and evaluation of assured protection of video data without overloading GWs or edge servers.

A high-level overview of the study is presented in Fig. 1, while Fig. 2 illustrates the physical testbed. In Fig. 1, the edge servers are equipped with hardware

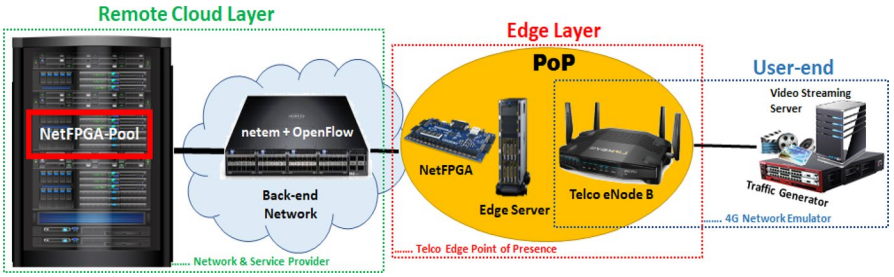

Fig. 2 Experimental testbed

Table 1 System components and their specifications

Traffic generator Ixia XM2	Edge server	NetFPGA-10G	Cloud (institutional-server)
Intel Pentium Mobile	Intel Core i7	Xilinx Virtex-5	Intel i7-3770
2 GHz	2.5 GHz, 4 cores	1 GHz, 2 Core	3.4 GHz, 20 cores
2 GB RAM 250 GB SATA	6 GB RAM DDR3	Two 4 GB DDR3 256 MB Strata Flash	32 GB RAM DDR3 2-TB HDD
Windows/Linux	Ubuntu 16.04.6 LTS	Fedora 14 (x86_64)	Ubuntu 16.04.6 LTS
2 × 10Gbps Ethernet	10 Gbps Ethernet	4 SPF Ports × 10 Gbps Ethernet	10 Gbps Ethernet

accelerators (HAs) comprising of FPGAs. The remote cloud site also has a pool of HAs, on which a variety of network functions and services can be boarded. This testbed allows us to investigate the impact and benefits of offloading VNFs from edge servers to FPGAs at both the edge and cloud as presented in Fig. 2. Users' privacy and security concerns are a significant obstacle deterring users allowing their devices to be participants in edge-cloud environments [20]. This testbed considers an encryption VNF case-study. Selection of this function was based on its significance to supporting user trust and data privacy as motivated in use cases.

4.1 Physical architecture and components

The testbed mirrors an edge telecommunication operator's infrastructure with wireless access to network services. In particular, the control and forwarding functions are decoupled using OpenFlow [21], offering many advantages over single vendor-based networking designs. It is worth mentioning that the user and edge setup (video generator and NetFPGA-equipped edge servers) were located in Dublin City University, Ireland, while the cloud part (pool of HAs and servers) was located in the University of Genoa, Italy. There are several components at each layer of the proposed approach as discussed in Sect. 4.1. The hardware and software specifications of components at three layers which are integrated in the experimental framework are detailed in Table 1.

Table 2 Video latency results

Test	Protocol	Chunk size	Mean execution time (ms)	Standard deviation (ms)	Mean E2E latency [ms]	Standard deviation (ms)
Case 1 (No-VNF)	TCP	Default	N/A	N/A	**29.73**	9.17
	UDP	Default	N/A	N/A	**21.80**	9.13
Case 2 (SW-VNF)	TCP	128	29.57	7.17	49.14	13.52
	TCP	256	27.38	6.34	46.56	12.62
	TCP	512	**25.44**	6.28	**44.24**	9.81
	UDP	128	27.95	5.48	47.38	8.63
	UDP	256	25.11	5.11	44.21	9.15
	UDP	512	**23.68**	5.09	**42.6**	11.71
Case 3 -(HW-VNF)	TCP	128	2.68	5.77	31.47	7.06
	TCP	256	2.06	3.93	30.66	8.28
	TCP	512	**1.91**	3.95	**28.20**	3.91
	UDP	128	2.18	4.58	30.8	6.19
	UDP	256	2.06	2.81	27.07	5.11
	UDP	512	**1.91**	2.54	**24.45**	2.37

The bold values reflect the time consumed as a performance indicator in terms of end-to-end delay and encryption VNF execution time

4.1.1 User-end

Video streaming devices are selected as a test application for experiments due to its stringent requirements. The 4 K video resolution with a frame rate of 30 fps using H.264 MPEG format is produced. A 4 K video chunk requires a data rate of $\approx$ 15Mbps for smooth streaming. An Ixia traffic generator is used to generate video streams. The video resolution is set to 4 K with a frame rate of 30 fps using H.264 JMPEG. Once a video stream is initiated, the video packets are sent to the edge server application using LTE connectivity through eNodeB which is realized using the Amari LTE 100 suite.

4.1.2 Edge layer

The edge server application reorganizes the incoming video stream into 128-, 256- or 512-byte chunks as configured in the test cases presented in Table 2. At this point, video chunks are forwarded to the NetFPGA, which is responsible for encrypting the incoming chunks. When encryption is finished, data are forwarded from the NetFPGA to the TCP or UDP port as per configuration. The edge server application serves as client and server for parsing the data, encrypting/decrypting and timestamping for latency measurements.

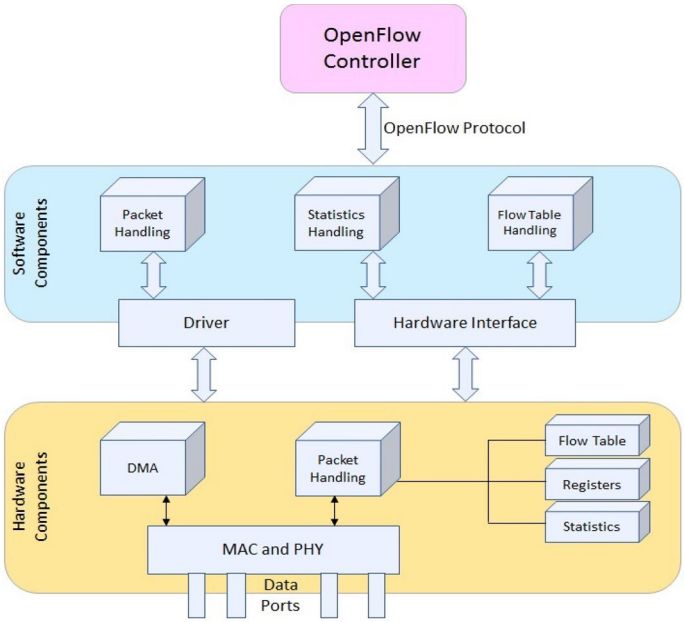

Fig. 3 OpenFlow implementation on NetFPGA-10G

The edge server is composed of a 4-core Intel i7 2.5 GHz (running Ubuntu 16.04.6 LTS) and 6 GB DDR3. It has two sub-components, namely an edge controller and an offloading module. The edge controller is implemented using OpenFlow to accept incoming video chunks and forward them to NetFPGA for VNF execution or to the remote cloud.

NetFPGA-10G [22] is based on Xilinx Virtex-5 (running Xilinx-Linux) and is connected via a 10Gbps on-chip bus. The NetFPGA logic area is programmed in Verilog for hardware implementation of the security VNF.

OpenFlow implementation in NetFPGA-10G In this work a NetFPGA-10G assisted OpenFlow switch[1] is extended to forward data packets as presented in Fig. 3. The edge controller module exchanges flow updates with the NetFPGA-10G OpenFlow controller. This is a programmable customized 4-port 10GE switch, which consists of a hardware implementation and associated software module. The software module is responsible for exchanging updates between edge server and NetFPGA OpenFlow controllers. The hardware component changes and forward packet from one port to other port(s), according to flowtable. The ports are configured for the following tasks.

Port 1: Incoming video chunks.

[1] https://github.com/NetFPGA/NetFPGA-public/wiki/NetFPGA-10G-OpenFlow-Switch.

Port 2: Encrypt/decrypt VNF (TCP).
Port 3: Encrypt/decrypt VNF (UDP).
Port 4: Outgoing video chunks.

Security VNF Encryption can offer a flexible way to protect data in transit for point-to-point and point-to-multipoint applications. In current real-time applications, traffic exchange among MEC and cloud is mostly unencrypted and subject to eavesdropping. Due to high computation involved in security protocols, lightweight symmetric ciphers have gained interest for data security in constrained computing environments. Considering the significance of information security, an encryption VNF is implemented in the proposed platform. We extended a lightweight symmetric block cipher, PRESENT [23]-based encryption/decryption on the NetFPGA-10G. PRESENT is implemented in two stages in NetFPGA. In the first stage, PRESENT is designed for a standard NetFPGA platform using Verilog, and in the second stage, the traditional FPGA implementation was imported to LabVIEW FPGA using the IPIN node. Data transfer between applications running on the real-time processor (edge server) and programs running on the FPGA (encryption/decryption) is based on writing data to controls and reading data from indicators using OpenFlow.

4.1.3 Cloud layer

The cloud layer consists of servers that represent a traditional cloud. It has Intel 3.4 GHz i7 processors with 20 Core CPUs and memory of 64 GB, which is more powerful than most virtual machines provided in commercial cloud services. We implemented the cloud layer using institutional servers at the University of Genoa in the H2020 INPUT[2] project. The servers have video applications to enable streaming, computing and storage. To consider the dynamic network conditions such as content location and load balancing, *netem* is configured for the default cloud settings [9].

5 Performance evaluation

5.1 Evaluation scenarios and metrics

The experiments replicate a setup of an end-to-end video stream from IoT devices to a remote cloud through an FPGA-enabled edge environment. The purpose of enabling encryption VNF at the edge is to provide the applications/devices with privacy, confidentiality and trust, while data are in transit through public networks. The experiments focus on metrics that are significant to time-sensitive video streaming applications, namely end-to-end latency, VNF execution time, and power consumption for the encryption VNF in the test cases. The latency measurements in all experiments have a mean value over 10 test runs with 95% confidence interval. The metrics are defined as following:

[2] https://www.input-project.eu/.

- *Execution time* is the time taken by the VNF module from starting to encrypt or decrypt data packets until it finishes.
- *End-to-end (E2E) latency* is the time from when data are sent by an IoT device until the data are received at the cloud. Latency results from several factors: packet processing, egress and ingress delays, packets queuing delay and transmission and propagation delays. However, the measurements presented here correspond to latency of the packets. This time includes encryption and decryption at the edge and cloud, respectively. A specification of 25 ms $<$ *low latency* $<$ 50 ms is a reference performance indicator for this work, as defined by the category 4 service requirement [24] for video surveillance, online auctions and live sports events.
- *Power consumption* is the power consumed by the VNF module while processing the data. We obtain the FPGA on-chip power consumption using the Xilinx Power Estimator (XPE) 2017.4. For the edge server power consumption, we use the *powertop* tool to measure the VNF energy consumption.

5.2 Evaluation results

A combination of options such as transport protocol, presence of encryption, and encryption chunk length was configured in the testbed. The motivation for such testing lies in the desire to be able to draw conclusions about the effect of video encryption on network performance; determine the parameters that result in optimal performance of encrypted video in terms of latency and execution delay; and compare performance with legacy approaches. For the purpose of result presentations, the experiments are divided into three test cases.

Case 1 (No-VNF) and Case 2 (SW-VNF) are baselines for this study versus Case 3 (HW-VNF). Case 2 and Case 3 consist of six experiments each, which were derived by using different combinations of options for the chunk size (128/256/512 bytes) and the transport protocol (TCP/UDP). The latency measurement results are taken using three sets of chunks considering AES for H.264 and MJPEG encoding scheme requirements. The encoding of the data was done for the camera video stream, while the stream was decoded in the latency measurement application utilizing the FFMPEG library. The data packets sent from the camera are accepted by the application and are reorganized into fixed size chunks of 128, 256 or 512 bytes, depending on data length encryption settings. For each case, two sets of results were obtained. One set represents execution time, while the other represents E2E latency.

Video latency measurements are presented in Table 2. It is noticeable that both Case 2 and Case 3 have higher mean values when compared to the Case 1 (No-VNF) due to the fact that there is additional data processing in VNF test cases. However, Case 3 stands out with mean latency for tests of both TCP and UDP being close to Case 1. This shows that HW-VNF implementation has minimal influence on the latency of video transmission in comparison with the software implementation in Case 2. The lowest secure video latency measured is in Case 3 (28.2 ms using TCP and 24.45 ms using UDP). The overall lowest latency for secure video was obtained using the combination of 512-byte chunks with UDP. An observation can be made

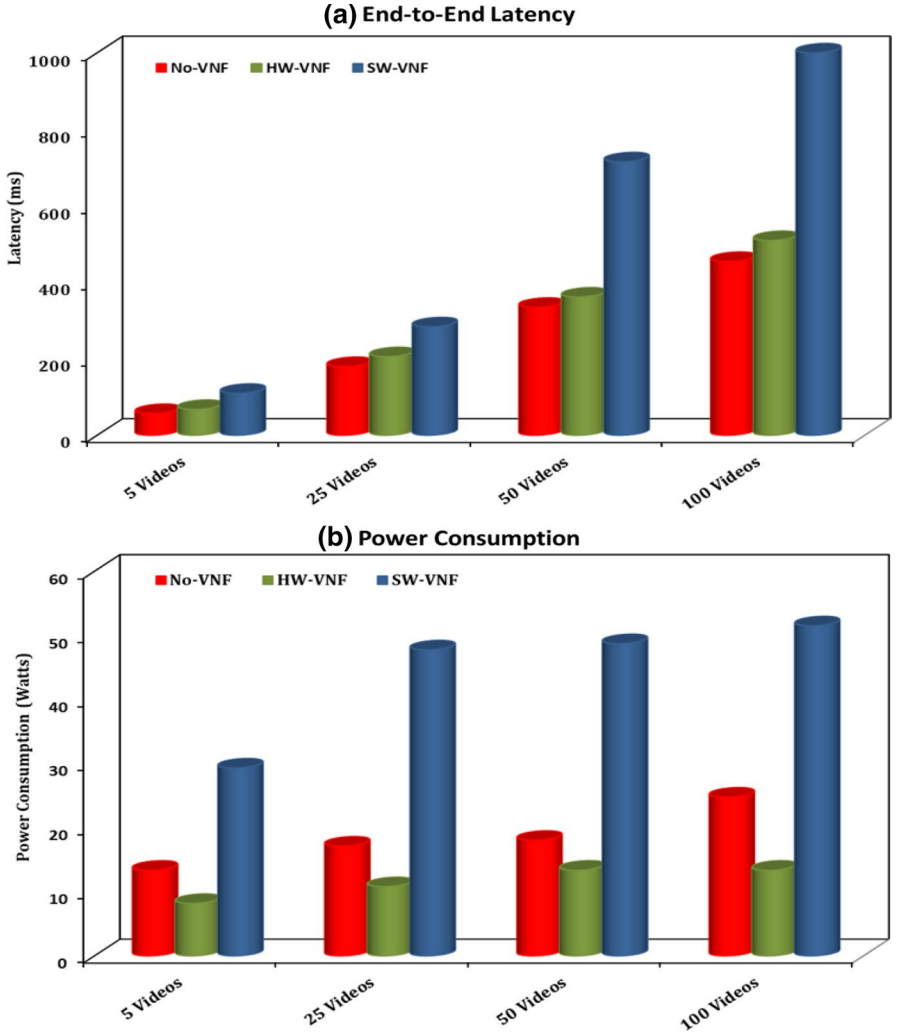

Fig. 4 Scalability performance evaluation: (**a**) E2E Latency (**b**) Power consumption

that cases with the TCP protocol result in higher latency than UDP due to difference in protocol implementations. Case 2 is conducted with software implementation of VNF resulting in $11x - 13\times$ higher execution time and 18–20 ms higher latency than Case 3 for a single stream.

Given encouraging results for a single stream, a further experimental campaign was performed to investigate the scalability of the HW-VNF implementation. A video stream with 5, 25, 50 and 100 videos is used for testing and evaluation representing four scales of transmission intensity (low, med, high and dense). Experimental results across the range of performance parameters are presented in Fig. 4.

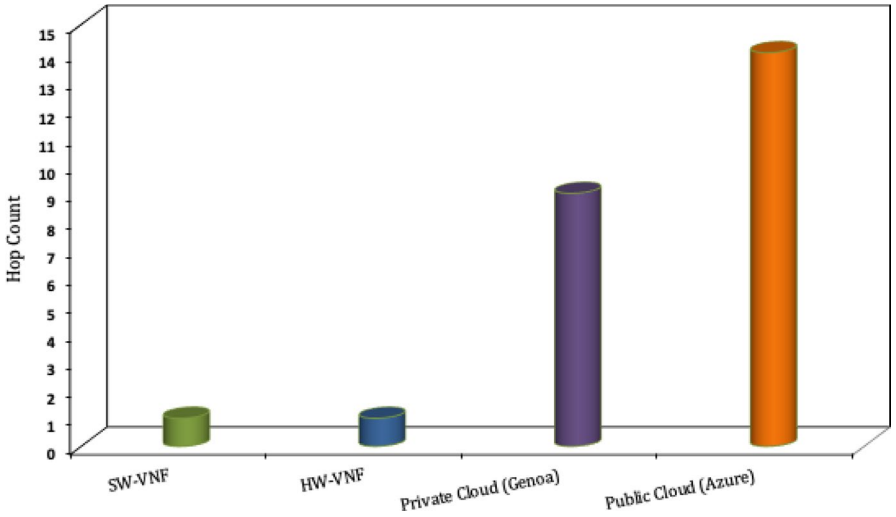

Fig. 5 Video transmisstion through unsecure hops

Based on the results shown in Table 2, the lowest latency combination of UDP and 512-byte chunks was used for these scalability experiments. The latency in securing and retrieving streamed video frames in Case 3 (HW-VNF) is approximately similar to Case 1 (No-VNF) for all scales as presented in Fig. 4a. The latency of Case 2 (SW-VNF) for low intensity is 76% and 86% higher than Case 3 and Case 1, respectively. At the dense intensity, Case 2 (SW-VNF) results in 77% higher latency compared to Case 3 (HW-VNF). In Fig. 4b, the power consumption for various video intensity levels is measured. Scaling the FPGA hardware frequency (between 63.5 and 125 MHz), using the frequency scaling technique developed in NetFPGA [25], the HW-VNF consistently consumes 72–76% lower energy than the SW-VNF at the edge server. The HW-VNF also consumes lower energy than Case 1 (No-VNF) because of the absence of adaptive transmission in the proprietary hardware.

In Fig. 5, the average hops for unsecured video transmission are evaluated for a public cloud (Microsoft Azure) and the private cloud. Both HW-VNF and SW-VNF have only 1 hop of unsecure transmission from end-user GW to cloud as compared to 9 and 14 hops, respectively.

6 Related work

We highlight how previous research combines EC and NFV, while lists of various edge approaches can be found in state-of-the-art surveys [26–28]. We present the related works that deploy VNFs at the edge in this section.

Authors present an architecture of a NFV-based MEC platform, and present an experimental study of the Quality of Experience (QoE) of HTTP videos deployed using such a platform [29]. Their findings confirm that MEC can further improve

the QoS of the application as opposed to running the same application on cloud/remote servers. However, their approach does not rely on any orchestration platform for deploying/managing VNFs for real-time applications. These functions are hard-coded on the machines, which do not allow for runtime changes. Such an approach cannot guarantee performance because of the missing resilience through an orchestration platform. Authors present a framework for on-the-fly transcoding in an edge environment [30]. The authors deploy a transcoding service as a VNF that can ensure dynamic rate switching of the streamed video. The authors report that the proposal can improve the QoE of users. However, the verification of this approach is made in a simulated environment. Also, like the above works, this approach does not rely on an orchestration platform to handle a compute-intensive task such as encryption at a very high data-rate. Researchers in [31] present an NFV-enabled MEC simulated framework to manage allocation of sufficient resources in order to guarantee continuous satisfaction of the application latency requirements. To demonstrate the usefulness of the proposed approach, a simulation-based performance evaluation of an augmented reality application with 1800 mobile users was carried out. Authors claimed that only their approach ensured MEC services respond to user requests on time.

Other researchers have focused on integrating GPUs with VNFs. In [32], researchers demonstrated the use of a general-purpose GPU in the context of VNFs for video transcoding. Zhi et al. [33] proposed a real-time distributed video transcoding system based on an edge-like environment of networked machines equipped with GPUs. The authors use Apache Storm [34], to ensure the communication of the video streams and to orchestrate the transcoding tasks in a group of networked physical machines. Work in [35] presents NetML, an NFV platform that allows the execution of machine learning algorithms in the edge server. NetML is built on top of OpenNetVM [36] NFV platform and uses GPUs to perform intensive ML tasks. The authors evaluated their work by calculating the throughput of an ML application performing object detection on streaming image data.

In a recent work [37], a mobile edge (ME) system is proposed that is based on the integration of a MEC architecture with an NFV framework. To demonstrate the viability, authors deployed and evaluated the performance of an ME application for CPU-intensive immersive video services. The edge infrastructure hosting the ME application exploits the use of GPUs to perform high CPU-intensive tasks. However, these works do not provide the resilience against information security for time-critical and continuous applications. Our work leverages NetFPGA as a networking device for the deployment of VNFs at the edge for the first time and conducts an investigation of its detailed performance based on time-sensitive application, in a real-time physical testbed.

7 Conclusion and research challenges

This paper evaluates the combined advantages arising from edge offloading using FPGAs at the network edge to accelerate VNF execution. The experimental results confirm the benefit of this approach over current edge counterparts by

offering data security (as a VNF) with 37–44% lower latency and 72–76% lower power consumption. While NetFPGA-based edge offloading is still in its early stage, we believe that this paper sheds lights on the potential to leverage new technologies to improve network services. Going further, placement/integration of programmable networked hardware at the local GW can provide end-to-end security requirements for sensitive and confidential data and zero trust.

Although, the realization of this approach is very promising, we must anticipate the inherent challenges due to the merger of these technologies.

- FPGA-based development is a complicated process requiring substantial FPGA engineering knowledge. Especially, when FPGAs are used with advanced edge middleware, web services and software systems, co-design is challenging. FPGA design is always a complicated task. The design process requires specific FPGA engineering knowledge, which remains a challenge for developers and usually results in a loss of productivity. Especially, when FPGAs are used together with advanced software systems become a real challenge.
- GPU is another widely used hardware. The GPUs can achieve higher throughput and peak speed is usually faster than FPGAs. In contrast, FPGA brings lower latency for single request, consume less energy and offers reconfigurability. The new FPGA's are in design phase to speed up the performance in comparison with latest GPUs
- The complexity of resource management grow manifold in the FPGAs not only due to virtualization of resources, but also due to requirements at different levels or domains such as network slice, network service and network resource (physical/virtual). Hence, a holistic resource management guided by defined policies is essential to ensure the appropriate management of FPGA resources.
- FPGA development, testing and debugging are more difficult than a software-only system. Current applications in the cloud are primarily software oriented.
- To enable a VNF-based open hardware platform, vendors must provide economical devices. Today, the cost of a high-end FPGA device is almost 10× of a Gigabit NIC. A higher price will ruin the business case for using FPGA technologies in VNF.

Funding Open Access funding provided by the IReL Consortium. This work was supported, in part, by the In-Network Programability for next-generation personal cloUd service supporT (INPUT) Project, EU Horizon 2020, under Grant 644672.

Data availability Private, H2020 INPUT members only.

Code availability https://github.com/NetFPGA/NetFPGA-public/wiki/NetFPGA-10G-OpenFlow-Switch

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical approval Not applicable.

Consent for publication Yes.

References

1. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: vision and challenges. IEEE IoTJ 3(5):637–646
2. Singh S, Chiu Y, Tsai Y, Yang J (2016) mobile edge fog computing in 5G Era: architecture and implementation. In: 2016 International Computer Symposium (ICS), Chiayi, pp 731–735
3. Hu Y-C, Patel M, Sabella D, Sprecher N, Young V (2015) Edge computing: a key technology towards 5G. ETSI White Paper 11(11):1–16
4. Schulz P et al (2017) Latency critical IoT applications in 5G: perspective on the design of radio interface and network architecture. In: IEEE Communications Magazine, vol 55, no 2, pp 70–78
5. Porambage P, Okwuibe J, Liyanage M, Ylianttila M, Taleb T (2018) Survey on multi-access edge computing for internet of things realization. IEEE Commun Surv Tutor 20(4):2961–2991
6. Qi H, Ayorinde OA, Calhoun B (2017) An ultra-low-power FPGA for IoT applications. In: 2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 1–3
7. Hauswald et al J (2015) Sirius: an open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In: ACM ASPLOS
8. M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing, vol. 8, no. 4, (2009).
9. W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen and P. &. S. M. Pillai, "Quantifying the Impact of Edge Computing on Mobile Applications," in APSys, (2016).
10. Zhao R, Song W, Zhang W, Xing T, Lin JH, Srivastava M, Gupta R, Zhang Z (2017) Accelerating Binarized Convolutional Neural Networks With Software-Programmable FPGAs. In: ACM FPGA
11. Putnam et al A (2014)A reconfigurable fabric for accelerating large-scale datacenter services. In: ACM/IEEE 41st ISCA, Minneapolis
12. Nurvitadhi E, Sheffield D, Sim J, Mishra A, Venkatesh G, Marr D (2016) Accelerating binarized neural networks: comparison of FPGA, CPU, GPU, and ASIC. In: 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, pp 77–84
13. Mohanty SP, Choppali U, Kougianos E (2016) Everything you wanted to know about smart cities: the internet of things is the backbone. IEEE Consum Electron Magaz 5(3):60–70
14. Mehmood Y, Ahmad F, Yaqoob I, Adnane A, Imran M, Guizani S (2017) Internet-of-things-based smart cities: recent advances and challenges. In: IEEE Communications Magazine, vol 55, no 9
15. York K (2016) DYN's DNS DDoS attack. https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/
16. Lom M, Pribyl O, Svitek M (2016) Industry 40 as a part of smart cities. In: 2016 Smart Cities Symposium Prague (SCSP), Prague, pp 1–6

17. Cardno A 6 Critical ideas behind the smart factory and internet of things. http://www.vizexplorer.com/6-critical-ideas-behind-the-smart-factory-and-internet-of-things-iot/
18. Eshratifar AE, Pedram M (2018) Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment. In: ACM Proceedings GLSVLSI, New York, USA, pp 111–116
19. Cisco Cisco visual networking index: forecast and methodology. https://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/visual-networkingindexvni/mobile-white-paper-c11520862.html. Accessed 26 Jul 2020
20. Farris I, Taleb T, Khettab Y, Song JS (2018) A survey on emerging SDN and NFV security mechanisms for IoT systems. In: IEEE Communications Surveys and Tutorials
21. Oda S, et al (2014) Flow-based routing schemes for minimizing network energy consumption using openflow. In: Proceedings of Energy
22. NetFPGA 10G specifications. https://netfpga.org/10G_specs.html. Accessed 26 Jul 2020
23. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJ, Seurin Y, Vikkelsoe C (2007) Present: an ultra-lightweight block cipher. CHES
24. Service requirements for the 5G system (2018) Stage-1 (release 16), 3GPP, TS 22.261, V16.6.0
25. Liu P, Chaudhry SR, Huang T, Wang X, Collier M (2019) Multi-factorial energy aware resource management in edge networks. IEEE Trans Green Commun Network 3(1):45–56
26. Mach P, Becvar Z (2017) Mobile edge computing: a survey on archi- tecture and computation offloading. IEEE Commun Surv Tutor 19(3):1628–1656
27. Dinh HT, Lee C, Niyato D, Wang P (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mobile Comput 13(18)
28. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor 19(4):2322–2358
29. Li S, Guo Z, Shou G, Hu Y, Li H (2016) QoE analysis of NFV—based mobile edge computing video application. In: 2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp 411–415
30. Dutta S, Taleb T, Frangoudis PA, Ksentini A (2016) On-the-Fly QoE-aware transcoding in the mobile edge. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp 1–6
31. Yang B, Chai WK, Pavlou G, Katsaros KV (20163) Seamless support of low latency mobile applications with nfv-enabled mobile edge- cloud. In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), pp 136–141
32. Comi P, Crosta PS, Beccari M, Paglierani P, Grossi G, Pedersini F, Petrini A (2016) Hardware-accelerated high-resolution video coding in virtual network functions. In: 2016 European Conference on Networks and Communications (EuCNC), pp 32–36
33. Chang ZH, Jong BF, Wong WJ, Dennis Wong ML (2016) Distributed video transcoding on a heterogeneous computing platform. In: 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp 444–447
34. Apache Storm (2019) http://storm.apache.org/
35. Dhakal A, Ramakrishnan KK (2019) NetML: an nfv platform with efficient support for machine learning applications. In: 2019 IEEE Conference on Network Softwarization (NetSoft)
36. Zhang W, Liu G, Zhang W, Shah N, Lopreiato P, Todeschi G, Ramakrishnan KK, Wood T (2016) Opennetvm: flexible, high performance nfv. In: 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pp 1–2
37. Cattaneo G, Giust F, Meani C, Munaretto D, Paglierani P (2018) Deploying CPU-intensive applications on MEC in NFV systems: the immersive video use case. Computers 7:55