LONDON SOUTH BANK UNIVERSITY

SCHOOL OF ENGINEERING

Dissertation

ENABLING ARTIFICIAL INTELLIGENCE ANALYTICS ON THE EDGE

by

VASILIS TSAKANIKAS

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

Approved by

Internal Examiner

Perry Xiao, PhD
Professor in the School of Engineering at London South Bank University

External Examiner

Nikolaos Thomos, PhD
Professor in the School of Computer Science and Electronic Engineering at the University of Essex

. . . pons asimorum . . .

Acknowledgments

I would like to express my deepest appreciation to **Prof. Tasos Dagiuklas**, who believed in me from the very first moment of my PhD project. His guidance and support acted as a beacon of knowledge and clear thinking when I needed them the most.

. . . to my brother **George**, who taught me which the important things in life are . . .

. . . to my sons, **Dimitris**, **Hermes** and **Philip**, with the promise to make up for the time I stole from them . . .

. . . above all, to my wife **Roula**, for being everything I dreamed for, . . . and so much more . . .

Vassilios D. Tsakanikas

ENABLING ARTIFICIAL INTELLIGENCE ANALYTICS ON THE EDGE

VASILIS TSAKANIKAS

London South Bank University, School of Engineering, 2022

Major Professor: Tasos Dagiuklas, PhD
Professor in the School of Engineering at London South Bank University

ABSTRACT

This thesis introduces a novel distributed model for handling in real-time, edge-based video analytics. The novelty of the model relies on decoupling and distributing the services into several decomposed functions, creating virtual function chains (VFC model). The model considers both computational and communication constraints. Theoretical, simulation and experimental results have shown that the VFC model can enable the support of heavy-load services to an edge environment while improving the footprint of the service compared to state-of-the art frameworks. In detail, results on the VFC model have shown that it can reduce the total edge cost, compared with a monolithic and a simple frame distribution models. For experimenting on a real-case scenario, a testbed edge environment has been developed, where the aforementioned models, as well as a general distribution framework (Apache Spark ©), have been deployed. A cloud service has also been considered. Experiments have shown that VFC can outperform all alternative approaches, by reducing operational cost and improving the QoS. Finally, a migration model, a caching model and a QoS monitoring service based on Long-Term-Short-Term models are introduced.

Contents

List of Tables

List of Figures

List of Abbreviations

1G	.....	1st Generation
2G	.....	2nd Generation
AHD	.....	Analog High Definition
AI	.....	Artificial Intelligence
AVE	.....	Augmented Virtual Environment
BDA	.....	Blu-ray Disc Association
BLE	.....	Low Energy Bluetooth
CCA	.....	Canonical Correlation Analysis
CCTV	.....	Close Circuit TV
CART	.....	Classication and Regression Tree
CNN	.....	Constitutional Neural Network
DL	.....	Deep Learning
FPS	.....	Frames per Second
HCI	.....	Human Computer Interaction
HDR	.....	High Definition Range
HMM	.....	Hidden Markov Model
HEVC	.....	High Efficiency Video Coding
LDA	.....	Linear Discriminant Analysis
LSTM	.....	Long Short-Term Memory
MEC	.....	Multi Access Edge Computing
ML	.....	Machine Learning
PCA	.....	Principal Component Analysis
PDF	.....	Probability Distribution Function
QoS	.....	Quality of Service
RNN	.....	Recurrent Neural Network
SFDM	.....	Simple Frame Distribution model
UVA	.....	Unmanned Aerial Vehicles
VF	.....	Virtual Function
VFC	.....	Unmanned Aerial Vehicles
VSaaA	.....	Video Surveillance as a Service
VSS	.....	Video Surveillance Systems
WAN	.....	Wide Area Network
ZR	.....	Zero Trust

Keywords: surveillance, computer vision, image analysis, analytics, image features, surveillance analytics, cloud computing, fog computing, edge computing

Chapter 1

Introduction

1.1 Background

The recent rise of a diverse set of video analytics, deep learning-based augmented and virtual reality applications drives the demand for real-time mobile cloud services. These real-time, mobile apps require heavy computing over big datasets and are often expected to deliver minimal end-to-end latency for acceptable end-user quality of experience. Limited battery life, computation and storage capacity constraints inherent to mobile devices mean that application executions must be offloaded to cloud servers, which then return processed results to the mobile devices through the Internet. End-to-end communication between cloud servers in remote data centers may result in long delays typical of multi-hop Internet communications.

To accommodate these rising edge applications, edge cloud computing has been presented as a potential solution (Bonomi et al., 2012), which brings computing to the network edge to reduce response time while avoiding edge-to-core network capacity limits.

1.2 Challenges and Motivation

Despite previous and ongoing work on many areas of edge computing, it is planned to approach this challenge in a fully latency-constrained architecture. The following issues arise as a result of this viewpoint.

First, how to optimally distribute real-time application functions across mobile devices, edge servers, and central clouds so that computing resources are fully utilized while providing global user scalability.

As a sample example, surveillance applications have been extensively accepted as a killer application in the era of edge computing. Edge cloud computing can help surveillance applications satisfy their QoS requirements, but merely relocating all surveillance tasks to the edge makes it more difficult for end users to have integrated services across the network since synchronizing users' profiles and video cameras across widely distributed edge clouds is a challenging task.

Second, the challenge of how to deploy these new edge applications efficiently within an edge cloud has not been thoroughly investigated. For edge applications, replicating the successful cloud computing concept can not apply. This is mostly due to edge clouds' very diverse characteristics. Edge clouds, unlike central clouds, are frequently made up of heterogeneous compute nodes with vastly disparate network bandwidths. For example, in (Zhang et al., 2017), the processing nodes and their interconnects are assumed to be generally homogeneous in central clouds, but in (Ha et al., 2017), the edge servers have significantly variable capabilities. As a result, managing these diverse resources efficiently in order to fulfill application latency limitations is a significant new research area.

Finally, current approaches only contemplate offloading jobs to a single server, assuming that the server has the resources to complete the tasks in a timely manner. In practice, however, a single edge server is generally outfitted with expensive hardware, such as Intel Xeon Scalable Processors with Intel Deep Learning Boost or NVIDIA EGX A100, that is shared by several clients (i.e., multi-tenant environment). Furthermore, resource fragmentation is caused by the varied resource demands of applications operating on edge servers and highly dynamic workloads by mobile users.

If the fragmentation isn't used effectively, it might waste a lot of resources throughout the edge servers.

1.3 Research objectives and contributions

This work introduces a novel distributed model for handling in real-time, edge-based video analytics, such as the ones required for smart video surveillance. The novelty of the model relies on decoupling and distributing the services into several decomposed functions (*Virtual Function* - VF) which are linked together, creating virtual function chains (Virtual Function Chain - VFC model).

The model considers both computational and communication constraints. Theoretical, simulation and experimental results have shown that the VFC model can enable the support of heavy-load services to an edge environment while improving the footprint of the service compared to state-of-the art frameworks.

For supporting the aforementioned approach, the following research questions governed the development of this work throughout its lifetime.

1. **RQ1**: Which is the most appropriate optimization model for VF placement in a heterogeneous edge/fog environment?

2. **RQ2**: Which is the most appropriate migration model for VF for load balancing and self-healing?

3. **RQ3**: Which is the most appropriate caching model for performance boosting of the proposed architecture?

As far as publishing the results of the work related to this PhD project, the following papers have been published or are being under review:

- (Tsakanikas and Dagiuklas, 2018) (Tsakanikas, V. and Dagiuklas, T. (2018). Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, 70:736-753.) reports the literature review which identified the current status of video surveillance systems, as well as the deployment technologies which nowadays support such applications are utilizing.

- (Tsakanikas and Dagiuklas, 2021) (Tsakanikas, V. and Dagiuklas, T. (2021). Enabling real-time AI edge video analytics. In *ICC 2021-IEEE International Conference on Communications*, pages 1-6.IEEE.) presented the proposed VFC model, along with the solution of the VF placement problem.

- (Tsakanikas and Dagiuklas, 2022b) (Tsakanikas, V. and Dagiuklas, T. (2022b). VFCSIM: A simulation framework of realtime multi-service virtual function chains deployment. In *GLOBECOM 2022 - IEEE Global Communications Conference* - IEEE.) details the architecture of the developed simulation environment for VFC modeling.

- (Tsakanikas and Dagiuklas, 2022a) (Tsakanikas, V. and Dagiuklas, T. (2022a). A generic framework for deploying video analytic services on the edge. *Transactions on Cloud Computing*, IEEE - R1 revision.) presents an extensive comparison of the proposed distribution model with similar approaches.

- (Tsakanikas, V. and Dagiuklas, T., VF Migration: A QoS aware migration model for edge deployed Virtual Functions) - to be submitted at *IEEE Transactions on Cloud Computing*. This paper presents the results reported in Chapter 5, regarding the proposed VF migration model.

1.4 Dissertation structure

The rest of this manuscript is structured as follows.

Chapter 2 presents an extensive literature review on surveillance systems, the technologies used nowadays to deploy them and the challenges such system are facing. The scope of this chapter is to identify the shortcomings of typical surveillance systems, as well as describing currently available edge deploying frameworks. Special focus is also placed on edge distributing technologies, as well as identifying and listing the challenges an edge network needs tackle, in terms of supporting demanding, real time services.

Chapter 3 introduces the ***Virtual Function Chaining*** model, while formulating and solving the VF placement problem. Additionally, comparison studies with similar approaches are presented in the same chapter. Chapter 3 constitutes the core of this thesis, as it describes the roadmap from the conception of the VFC model, to its formulation and its materialization and evaluation. The mathematical formulation of the model, as well as the methodology for tackling the VF placement problem are also detailed in Chapter 3.

Chapter 4 presents the architecture of $VFCSIM$, a simulation environment for VFC modeling. $VFCSIM$ is an open source simulator, designed and built within the work of this PhD project. Its scope is to facilitate the evaluation of the proposed model against other distribution approaches. The relative results of this chapter promotes the usage of the $VFCSIM$ simulator for simulation of virtual function chain scenarios.

Chapter 5 introduces, details and evaluates the VFC migration model for boosting quality of service and supporting self-healing mechanisms. After formulating the migration problem, appropriate solutions and algorithms are discussed. Finally, the VFC migration model is presented and its performance is evaluated through simulation scenarios on the $VFCSIM$ engine.

Finally, **Chapter 6** discusses the conclusions which can be drawn from the manuscript

and outlines the related future work.

Appendix A outlines the main mathematical principles for optimization theory.

Chapter 2

State of the Art

This chapter presents the current status of video surveillance systems. The main components of a surveillance system are presented and studied thoroughly. Algorithms for image enhancement, object detection, object tracking, object recognition and item re-identification are presented. The most common modalities utilized by surveillance systems are discussed, putting emphasis on video, in terms of available resolutions and new imaging approaches, like High Dynamic Range video. The most important features and analytics are presented, along with the most common approaches for image / video quality enhancement. Distributed computational infrastructures are discussed (cloud, fog and edge computing), describing the advantages and disadvantages of each approach. Additionally, deep learning algorithms are presented, along with the smart analytics that they utilize. Augmented reality and the role it can play to a surveillance system is reported, just before discussing the challenges and the future trends of surveillance.

2.1 Introduction

During the past decade Video Surveillance Systems (VSS) have revolved from simple video acquisition and display systems to intelligent (semi)autonomous systems, capable of performing complex procedures. Nowadays, a VSS can integrate some of the most sophisticated image and video analysis algorithms from research areas such as classification (e.g., neural networks or stochastic models), pattern recog-

nition, decision-making, image enhancement and several others. Thus, a modern surveillance system comprises image and video acquisition devices, data processing - analysis modules and storage units, components, which are all crucial for the system's workflow.

Surveillance systems have technically evolved under three generations. The 1st generation (1G) is dated back in 1960's, when analog Close Circuit TV (CCTV) systems were first introduced, mainly for indoor surveillance applications. For that time, 1G systems performed rather satisfying, gaining the trust of the market with deployments in banks, supermarkets, garages, etc. Yet, analogue technology constrained their capabilities, especially for recording and distributing processes. In 1980's, digital imaging evolved surveillance systems to the 2nd generation (2G), offering two major advances. First, compression (Galteri et al., 2018) and distribution (Qiu et al., 2022) have now become more efficient and more cost-effective. Second, computer vision algorithms have been introduced to surveillance systems, offering semi-automated functionalities, such as object tracking and event alerting. Finally, since the early 2000's, one can discuss about the 3rd generation of surveillance systems, where fully automated wide-area surveillance systems are explored, aiming to offer reasoning frameworks and behavioral analysis functionalities, incorporating and integrating at the same time multi-sensor platforms and data fusion techniques. In Fig. 2·1, a timeline diagram of the evolution of surveillance systems is depicted.

There are many flavors of Video Surveillance Systems, each one trying to fulfill part of the market. Several categorizations can be drawn. Hence, one can categorize video surveillance systems based on the type of imaging modality acquired, producing categories like *"one camera systems"*, *"many camera systems"*, *"fixed camera systems"*, *"moving camera systems"* and *"hybrid camera systems"*. Another categorization can be based on the applications which a Video Surveillance System offers, such as object

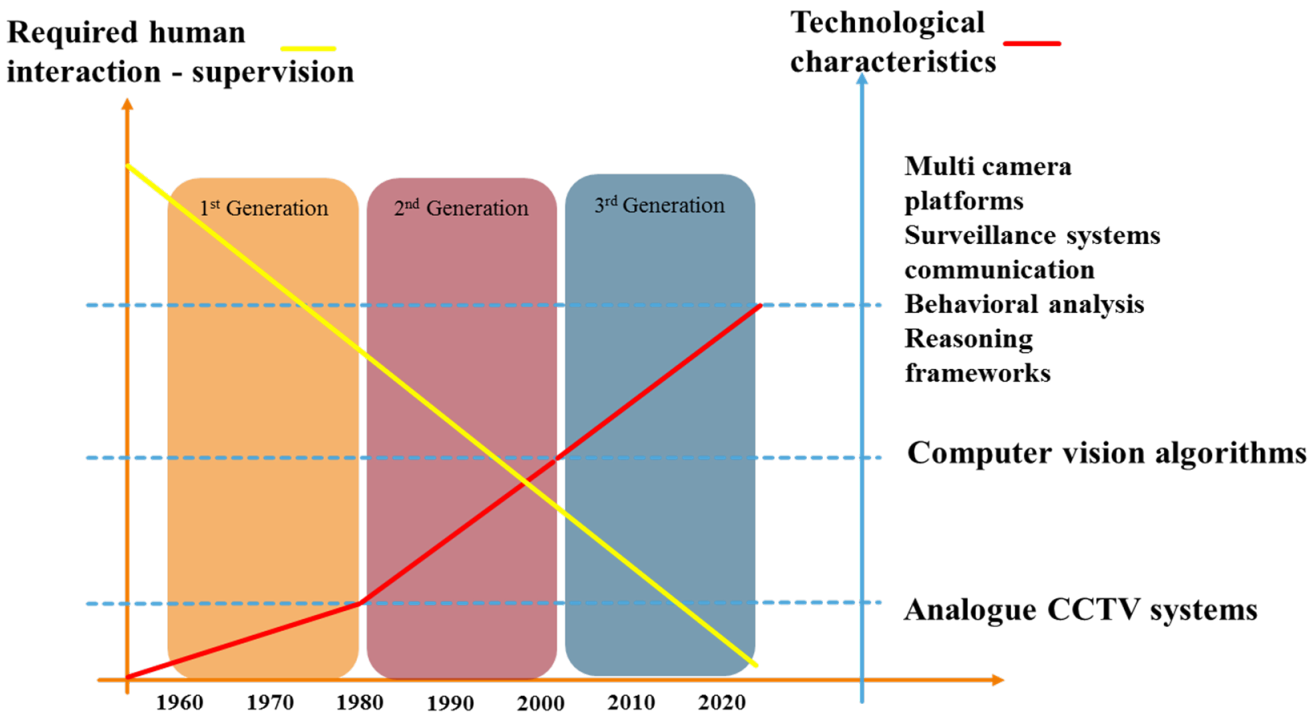

Figure 2·1: Evolution of surveillance systems.

tracking, object recognition, ID Re-identification, customized event alerting, behavior analysis etc. Finally, Video Surveillance Systems can be categorized based on architecture a system is built on, such as stand-alone systems, cloud-aware systems and distributed systems.

For most of the time, surveillance systems have been passive and limited in scope. In this context, fixed cameras and other sensing devices such as security alarms have been used. These systems are able to either track persons or to detect some kind of events (e.g., a person breaking the door or the window), however, they have not been designed to predict abnormal behaviors (Tsushita and Zin, 2018). During the last years, there was a huge progress in sensing devices, wireless broadband technologies, high-definition cameras, and data classification and analysis. Combining such technologies in an appropriate way will allow to develop new solutions that extend the surveillance scope of the current systems and improve their efficiency. Within the

context of surveillance systems, efficiency improvement has two directions.

- First, the improvement of the video processing algorithms along with the derived video analytics will increase the validity and the accuracy of a surveillance system and
- second the integration of surveillance systems with cloud infrastructures is expected to improve reliability (e.g., generate alarms under poor lighting conditions etc.), reduce the maintenance costs and increase the response time of the systems.

VSS have to cope with several challenges, including, but not limited to, algorithmic and infrastructure ones. Thus, surveillance systems have to adapt with the emerging network and infrastructure technologies, such as cloud systems, in order to provide more robust and reliable services. This trend will also demand the integration of different surveillance systems for extracting more useful knowledge. This integration will require new communication protocols and data formats between surveillance agents, as well as new surveillance adapted databases and query languages. Finally, more accurate algorithms are required, especially in the context of behavioral analysis and abnormal activities detection.

The scope of this chapter is to survey the current status of VSS, aiming to identify the best practices for image and video processing and analysis and highlights research challenges for next generated systems. Additionally, the applicability of proposed algorithms and architectures will be assessed, in terms of time response and scenarios variety. The chapter is structured as follows: Within section 2.2, the available video sensors from different surveillance systems are presented, Section 3 describes the different modalities that are commonly used in surveillance systems. In Sections 4 and 5 several approaches for the most studied image and video processing algorithms

are analyzed, focusing on video analytics and quality enhancement respectively. In Section 6, computing architectures for boosting the performance of a surveillance system are discussed while in Section 7 the future trends on surveillance systems are drawn.

2.2 Video sensors

Nowadays, there is a variety of video sensors used from surveillance systems (Prati et al., 2019). As the technical specifications of the video sensors play a key role to the potential of a surveillance system, in this section an outline of the sensors' technical characteristics is provided.

The oldest and most used type of video sensors is analog video sensors which are used to CCTV surveillance systems. The resolution of the analog cameras is measured in vertical and horizontal line dimensions and typically limited by the capabilities of both the camera and the recorder that the CCTV system is using. In Table 2.1, common formats of analog cameras are provided, along with their resolution are presented. Until 2017, the higher resolution for analog systems came from the D1 format. Yet, since 2015 the AHD CCTV (Analog High Definition) cameras were introduced in the market, along with the corresponding recorders. Regarding the FPS (Frames per Second), specification of analog video sensors, it can vary from 1 FPS to 30 FPS. The majority of the systems use either 15 FPS or 7.5 FPS, as higher values require a large amount of storage volume, in case of recording.

Table 2.1: Resolutions of common analog video cameras.

Analog Video format	**Resolution**
1,080p Resolution	1,920 x 1,080
720p Resolution	1,280 x 720
D1 Resolution	704 x 480 (NTSC for the United States) 720 x 576 (PAL for Europe)
CIF Resolution	352 x 240
QCIF Resolution	176 x 120

During the last fifteen years digital video sensors gained their market share against analog technology. While analog sensors transmit the captured data uncompressed, the digital sensors perform digitalization of the input stream and thus can take advantage of compressing algorithms and advance video codecs. Consequentially, these sensors can interface directly with network infrastructures and transmit their data over switches and routers. This is the reason why the digital sensors often referred as IP cameras. The resolution and the frame rate of digital sensors are adjustable. Common IP-based cameras, which nowadays belong to the HD (High Definition) category can capture video on 1,920 x 1,080 resolution and 30 FPS and downgrade to 1,280 x 720 or D1 for 15 FPS. Ultra HD (UHD) video sensors have been also introduced to surveillance systems, pushing the available resolutions to 4K (3,840 x 2,160, usually under15 FPS) or 2,048x1,536 under 30 FPS.

Finally, since the beginning of 2010, a new type of video cameras has been introduced, the High Dynamic Range (HDR) video sensors. These sensors, which usually operate at HD resolution, are able of capturing the same scene multiple times using different exposure times (the time interval the camera shutter remains open and collects data) and then combine these frames to a single image. This technique, which nowadays is available only to high-end video cameras, makes the bright areas of the scene darker and the dark areas brighter, enhancing the quality of the video stream. HDR cameras (as well as HD and UHD cameras) utilize the H.264 video codec. Additionally, the research community, during the last few years has proposed the usage of High Efficiency Video Coding (HEVC) as an appropriate video coding standard for HDR content. Recently, several organizations including the Blu-ray Disc Association (BDA), the High-Definition Multimedia Interface (HDMI) Forum, and the Ultra-High Definition (UHD) Alliance have decided to adopt a delivery format based on HEVC Main 10, commonly referred to as *'HDR10'*, for the compression and delivery HDR

content.

2.3 Acquired modalities

All VSS utilize video streams. Yet, this is not the only modality a surveillance system can use. In this section, a brief description of systems utilizing additional modalities is provided. This section, presents the state of the art in surveillance system with respect to different modalities.

2.3.1 Sound

The most common modality to couple with video in a surveillance system is sound. There are two types of audio-visual data fusion architectures. In the first type, audio data are spatialized utilizing microphone arrays, aiming to improve tracking algorithms while in the second type, which is more general, sound is captured using a single microphone.

The most usual scenario for the first type of the systems is a known environment (indoor in the most cases) which is equipped with fixed cameras and microphones. For example, in (Zotkin et al., 2002), moving objects are located calculating the sound time delays among the microphones. Applications utilizing sound as modality are multi-object 3D tracking and walking person detection. These approaches include audio source separation, dynamic Bayes networks, learning and interference of graphical model and 2 - layer HMM (Hidden Markov Model) frameworks.

As for the second type of fusion architectures, due to the presence of only one microphone, audio spatialization in no longer available. Hence, the most common approach for audio-visual fusion is Canonical Correlation Analysis (CCA), using as variables spectral bands for sound and image pixels for video. One of the main drawbacks of CCA is the need of large amount of data for model training. Some research works try to tackle this issue, like (Zou and Bhanu, 2005), in which a presumed

sparsity of the audio-visual events is exploited. Other approaches for audio-video correlation are proposed in literature. According to these approaches, two groups of multi-variate variables are correlated using the MMI (Maximization Mutual Information) method, while Markov chains are proposed and the audio-video joint densities are estimated using a group of training sequences.

2.3.2 GPS

Video surveillance systems started to incorporate GPS data when they stopped using fixed cameras and started to incorporate moving cameras. This required addition of an extra layer of meta-data to the tracking algorithms. Yet, the raising interest for aerial video surveillance systems led to the design of surveillance architectures, which incorporated moving cameras installed either on drones or on UAV (Unmanned Aerial Vehicles). One of the first research works, which proposed a surveillance system with moving cameras was (Kumar et al., 2001), where a framework for real-time, automatic exploitation of aerial video for surveillance applications is presented. The main functionality of the proposed system is performed by a module, which separates an aerial video into its natural components, namely the static background geometry, moving objects and appearance of the static and dynamic components of the scene. The system finally attempts to register the geo-location of video with the tracked objects, using GPS data and elevation maps before producing re-projected mosaics of the scenes.

Besides utilization of GPS data from UAV surveillance systems, geo-location is also used from in-vehicle surveillance systems. Systems under this framework have been proposed many research works. The basic idea behind these systems, is the registration of the tracked objects with the GPS data, in order to facilitate the creation of a meta-data map with of the trajectories of the tracking objects.

2.3.3 Video

Undoubtedly, video streams are the primarily modality when it comes to surveillance systems. At some level, most of the research works regarding surveillance systems try to mimic the biological process of how people detect events and categorize them. For example, a common pre-processing procedure of event detection algorithms is background / foreground classification, where the system tries to distinguish the static scene (which usually has no interest) from the dynamic foreground objects. This procedure is similar to the bioprocess where neurons detect a change in luminance and color of neighboring points after a short delay.

The quality of the acquired video stream plays a key role to the potentials of a surveillance system. Resolution, frame rate per second and contrast are some of the most important features of a video sensor. For example, a high quality video sensor can substitute a pre-processing enhancement algorithm, boosting up the response time of a surveillance system. On the other hand, usage of high resolutions results to increment of bandwidth requirements for data transmission and storage.

2.3.4 Modality fusion and intelligent surveillance systems

Data fusion is the process of combining two or more modalities in order to acquire more efficient and useful information compared to the acquired information when using the modalities separately. The concept of data fusion is not new, however, merging different types of data generated by heterogeneous devices is still a challenge. In the literature, different approaches to deal with this problem have been proposed. Statistical analysis where typical techniques such as mean, median, standard deviation, and variance (including Kalman filtering) are used is the straightforward approach. Most of the data fusion being used now rely on probabilistic descriptions of observations and use Bayesian networks to manage the uncertainty and combine this information.

In this category, one can also mention the techniques based on fuzzing and Dempster-Shafer theory, and learning algorithms based on neural networks and hybrid systems. The approach to be used often depends on the type of data, the level of reliability foreseen, and the requirements of the application (in our case the intelligent surveillance). Finally, sensor fusion and thermal–visible video registration techniques are proposed in (Torabi et al., 2012), where sensor fusion uses aligned images to compute sum-rule silhouettes, and then constructs thermal–visible object models.

2.4 Knowledge extraction algorithms

Within this section, focus will be given on the modules of a surveillance system, which are responsible for "translating" the raw video data to specific structured information. The most common activities on this field are face detection, face recognition, object re-identification and object tracking.

2.4.1 Face detection

Detecting faces within a scene is a mature problem in the area of computer vision. This is because face detection is one of the most widely used processes within surveillance systems, as it is required by many applications such face recognition, face tracking, face analysis for behavioral knowledge extraction (Zafeiriou et al., 2015). Yet, new applications constantly emerge, such as Human Computer Interaction (HCI), which demands more robust and accurate solutions.

The aim of face detection is to firstly to determine whether any faces are depicted in a scene and secondly to calculate and return the coordinates of the detected faces. This task involves many non-trivial conditions, such as variations in scale, location, orientation and pose, as well as lighting conditions, facial expressions and occlusions. One common classification of face detection approaches is reported in (Yang et al., 2002), where four categories are described: (a) template matching methods, where

pre-stored face templates are used to decide whether an image contains a face or not, (b) knowledge-based methods, where well established pre-defined rules are used, (c) feature invariant approaches, where structural face features are utilized and (d) appearance-based methods, where models are trained against annotated face data. Nonetheless, (Zafeiriou et al., 2015) suggests that the innovative work of Viola and Jones ((Viola and Jones, 2001)) has changed the way modern approaches for face detection are classified, and suggests that face detection algorithms should be categorized to algorithms that are based on rigid-templates and to algorithms that deploy Deformable Parts-based Model in order to model potential deformations among facial points.

One of the most important representative of the rigid-templates category is the work reported in (Viola and Jones, 2001). Within this work, Viola and Jones proposed a face detector which is based on the integral image, classifier learning with AdaBoost and the attentional cascade structure. Following this concept, new image features have been proposed in order to improve the accuracy of the algorithms. Such features are joint Haar features, which are based on the co-occurrence of multiple Haar-like features and Classification and Regression Tree (CART) based weak classifiers. Another common feature for face detection is based on regional statistics such as histograms, with Histogram of Oriented Gradients (HOG) being the most popular one. Lately, an approach that uses the so-called Integral Channel Features (ICF) with boosting achieved state-of-the-art performance in face detection under various conditions. Regarding the classification schemes, neural networks are widely used, like constrained generative model (an auto-associative, fully connected multi-layer perceptron with three large layers of weight) and convolutional neural network (CNN) based approaches.

As far as Deformable Parts-Models (also known as pictorial structures modelling)

is concerned, they constitute one of the standard choices for developing generic object detectors. While simple models have been proposed, more complex approaches have provided robust solutions.

2.4.2 Face recognition

Face recognition constitutes the problem of recognizing a face against a predefined knowledge database of faces. Face recognition problem implies that a face is already detected in a scene, which makes face detection a prerequisite process for face recognition. This problem troubles researchers for more than forty years, trying to produce robust, accurate and real-time solutions. The first approaches documented tried to model the face recognition problem as a two-dimensional pattern recognition problem, calculating "important" distances of facial features, such as the distance between the eyes of the length of the lips.

Nowadays, one can classify the methods for face recognition in three categories; namely holistic matching methods, feature-based methods and hybrid methods. Holistic methods suggest that the whole face region is compared against a face database using specific techniques such as Eigenfaces, Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Feature based methods are trying to extract facial geometrical features, such as mouth, lips, nose and eyes. These features are used as an input to classifiers, aiming to detect the match closest to the face detected. Feature based methods need to reformulate in order to produce accurate results when the aforementioned features are not visible in the scene. In order to tackle this problem, feature estimation methods have been proposed, mainly taking advantage of face structural constraints. For example, Ahonen *et al.* (Ahonen et al., 2004) proposed a novel approach for face recognition which incorporates both texture and shape information to represent faces. A face is first divided into small blocks from which the Local Binary Pattern (LBP) features are extracted and united into a

single feature histogram, which represents the face. In a more recent approach, Lenc and Král (Lenc and Král, 2016) propose to use dynamic positions and number of the fiducial points produced by LBP features. The selection of the chosen fiducial points is performed fully automatically, utilizing a set of Gabor filters. Local extrema in the filter responses are detected and used as the feature points. The number of points is further reduced using the K-means clustering algorithm. Finally, hybrid methods take advantage of both the techniques of holistic and feature based methods. These methods use as input 3D images and for that they can use information concerning the forehead or the chin shape.

During the past few years, face recognition algorithms have come to a maturity level where they can be used on real-world applications and uncontrolled environments. This fact brought up the need for developing new approaches in face recognition problem, such as the *"watch-list"* problem. According to this problem formulation, the system needs to distinguish among a very large number or individuals only the people who belong in a predefined list. A research work which tries to address this problem can be found in (Kamgar-Parsi et al., 2011), where for each individual in the watch-list, a classifier is trained. Then, for the detected individuals, certain features are used as input to the classifiers, reaching to the final decision.

2.4.3 ID Re-identification

The ID re-identification problem appears on multi-camera surveillance system setups, where people walk around the view field of numerous cameras (e.g. the scene of Fig. 2·2). Within such setups, a surveillance system should have the ability to track people across multiple cameras, thus performing crowd movement analysis and activity detection. More specifically, given a video of a person taken from one camera, re-identification is the procedure of identifying the person from videos taken from different cameras. Re-identification is crucial in establishing reliable individuals tagging

across multiple cameras or even within the same camera, when discontinuities and "blind" spots appear.

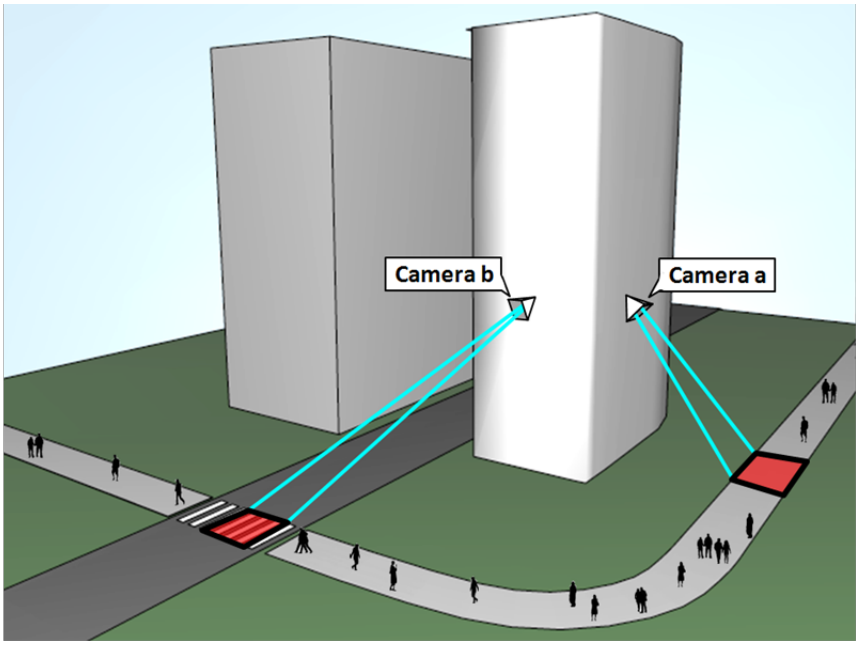

Figure 2·2: Person re-identification scenario.

ID re-identification is a challenging problem due to the visual vagueness and spatiotemporal uncertainty in a person's appearance across different cameras. These difficulties are often reinforced either by low-resolution images or poor quality video streams. Issues like these forced the research community to put focus on the ID-identification problem during the last years, aiming to produce robust and wide-applicable algorithms.

Since 2010, there has been many research works, which tried to address the ID re-identification problem. The problem of ID re-identification has been modeled as recognition problem. Given an image (or images) of an unknown person and a database of known people, the scope is to produce a sorted list of all the people in the database based on their similarity with the unknown individual. Thus, it is expected that the highest ranked match in the database will provide an ID for the

unknown person, thereby identifying the probe. In this scenario, it is assumed that the unknown person is included in the database of known persons (closed-set ID re-identification). Most of the approaches nowadays are based on appearance based similarity features between frames to establish common similarities. Typical features used to quantify individual similarities are low-level color features and texture features based on clothing. Nonetheless, such similarity features are only valid for a short period of time as people dress differently from day to day, or even through the same day. Hence, similarity based features are only suitable for a short period of time (short-term re-identification), which is the version of re-identification problem the research community mainly tries to solve. The methods and the techniques for ID re-identification are categorized in several methods, as depicted in Fig. 2·3.

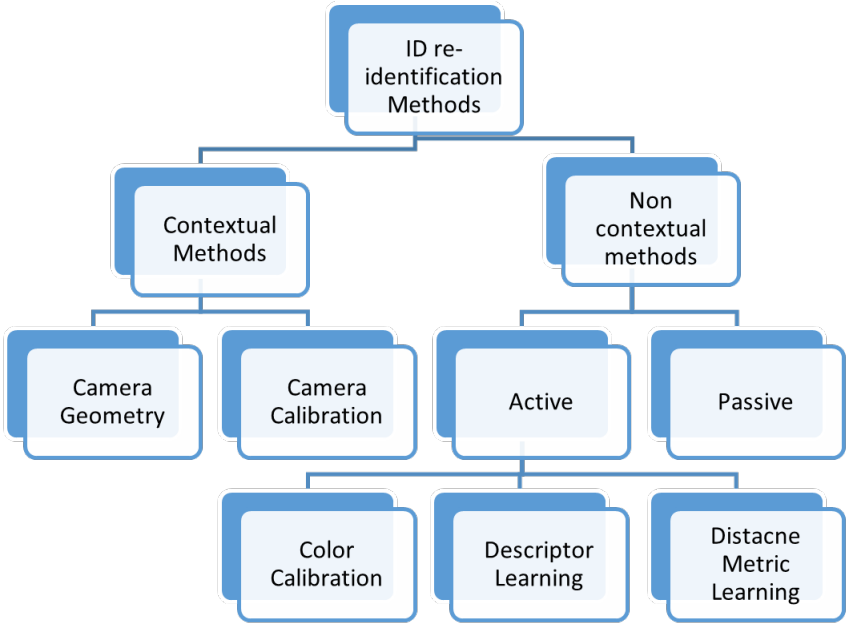

Figure 2·3: ID re-identification methods categorization..

Contextual methods

Contextual methods take advantage of external information such as camera geometry and camera calibration. For example, camera geometry setup is taken into account in

order establish intra-camera relationship and increase constraints among the cameras.

Camera geometry is usually determined by correlating activities among cameras with disjoint field of views and do not rely on information from tracking algorithms. The time-delayed correlations of activities are observed and quantified, utilizing multiple camera views in a single common reference space. Then, the assessment of the time delayed motion correlations is used for person re-identification and both temporal and spatial topology inference of a camera network. As far as the camera calibration as context concerns, camera field of view information and homography are considered, aiming to extract features from visual descriptors. For example, in (Lantagne et al., 2003), individuals' height is calculated using homography, to estimate a 3D model. A Panoramic Appearance Map (PAM), uses information from multiple cameras that view the object to produce a single object signature. Other important works in this category are reported in (Baltieri et al., 2011). Hu *et al.* proposed a method for people tracking using multiple cameras based on the detection of principal axis for each tracking person, which are the perpendicular segments from head to toe and from shoulder to shoulder. The algorithm estimates the principle axis for each camera and then attempts to correspond them in order to re-identify people. A modelling approach is also proposed in the literature, where 3D information is extracted from multiple cameras. The proposed model is a 3D Marked Point Process model using two pixel-level features. The workflow includes the feature extraction from multi-plane projections of binary foreground masks and the statistical estimation of the height and the position of each person. Finally, a 3D body model based long-term tracking algorithm connects missing or hidden tracks and is used to re-identify people.

Non-Contextual methods

Non-contextual methods rely on knowledge extraction using only the video stream as input, ignoring the contextual data. These methods, which are reported with a high frequency during the past years, are further categorized to both passive and active. Passive methods extract visual features in order to classify an individual's appearance against a known dataset (the description passive comes from the fact that these methods do not use machine learning techniques for feature extraction). Shape and color visual features for person modeling is proposed in the literature, where the video stream is divided in polar bins and Gaussian model along with edge pixels from each bin are used to produce the features. On the same page, a spatio-temporal segmentation method, utilizing watershed segmentation has been used, where the appearance of an individual is a combination of color and edge histograms.

On the other hand, active methods utilize machine learning algorithms for feature extraction. A machine-learning algorithm can either be supervised or unsupervised. The supervised approaches require a set of annotated training data, in order to "learn" to detect the desirable features (e.g. person's silhouette), while the unsupervised algorithms utilize clustering techniques in order to estimate different image features (without use of training data). One can categorize these machine learning methods into three categories; namely distance metric learning methods, descriptor learning and calibration methods. Distance metric learning methods do not use feature selection techniques for feeding learning algorithms. Yet, they place effort on learning suitable distance metrics, which are able to maximize the accuracy of the classification, regardless of the choice of appearance representation.

The descriptor learning methods try to acquire the most discriminative features in order to achieve ID re-identification. Another approach is to deploy a learning phase in order to produce descriptive lists of features that better represent an individual's ap-

pearance using a bag-of-features approach. Within such approaches, co-occurrences between a priori learned shape and appearance features produce an individual descriptor. HOG (Histogram of Oriented Gradients) features are also utilized by many research works. Finally, the color calibration methods try to model the color relationships between a specific pair of cameras using color calibration techniques. They usually employ a learning phase to produce the calibration model.

2.4.4 Object detection and tracking

Object detection and object tracking are the most common applications on video surveillance systems. Object detection constitutes the problem of isolating a specific region of a video stream based on the system's parameters while object tracking is a process of keeping track of the aforementioned region's motion. One can classify the object detection algorithms in four categories; namely Background Subtraction, Temporal Differencing, Frame Differencing and Optical Flow.

Algorithms using background techniques try to separate foreground objects from the background of the scene. In order to achieve this, background modelling (reference model) is mandatory. The more accurate and adaptive the background model is, the more accurate the detection algorithm is. The most common techniques to achieve background modeling include median and mead filters.

Temporal Differencing algorithms calculate the difference (on pixel level) between successive video frames, in order to detect the moving object. These algorithms are able to quickly adapt to highly dynamic scene changes. Yet, they suffer from important drawbacks; the most important of them is detection loss when the object stops moving and when the object's color texture is similar to the scene (camouflage). Also, false object detection may occur when scene objects tend to move (e.g. leaves of a tree when the air is blowing).

A simple approach of temporal differencing is Frame Differencing, where the tem-

poral information indicates the moving objects of the scene. In such methods, presence of mobility is established by calculating the difference (pixel level) of two successive video frames. Finally, Optical flow is the pattern of objects motion in a visual scene caused by the relative motion between an observer and the scene. Optical flow methods use partial derivatives with respect to the spatial and temporal coordinates in order to calculate the motion between two image frames. Such methods seem to be more accurate that the aforementioned approaches, but due to the computational time required and the noise tolerance, they are unsuitable for real (or near real) time scenarios.

Regarding the object tracking algorithms, their scope is to return the route for an object by calculating its relative position for each video frame. Object tracking can be classified as kernel based tracking, point based tracking and silhouette based tracking (Athanesious and Suresh, 2012) (Fig. 2·4).

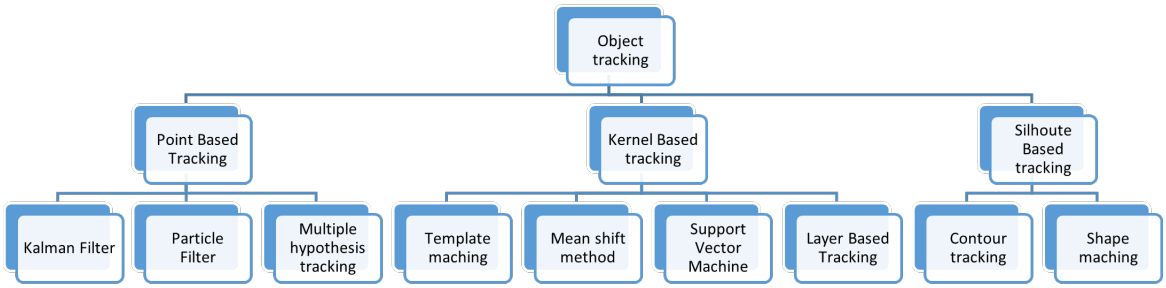

Figure 2·4: Object tracking methods.

The most common point-based approaches utilize Kalman and Particle filters. Kalman filter is a set of equations that provide recursive computational means to estimate a process's past, presence and future. Methods utilizing Kalman filter are based on Optimal Recursive Data Processing Algorithm. On the other hand, Particle Filter generates all models for one variable (e.g., contours, color features, or texture mapping). The particle filter is actually a Bayesian sequential importance technique.

In Multiple Hypothesis Tracking algorithm, several frames are observed for better tracking outcomes (iteration algorithm). Each hypothesis is a crew of disconnect tracks and for each hypothesis, an estimation of object's position in the following frame is made. The predictions are then compared by calculating a distance measure, allowing multiple hypothesis-tracking algorithms to track multiple objects.

In Kernel based tracking, kernel denotes to the object representations of rectangular or ellipsoidal shape and object appearance. The objects are tracked by estimating the movement of the kernel on each successive frame. Kernel based approaches can be classified in four categories. Template matching algorithms employ a brute force method for examination of the video frame, aiming to detect the region of interest. In template matching, a reference image is verified with the frame that is separated from the video. Template matching algorithms are able to detect small pieces of a reference image, but the usually work for only one object and they require computational heavy load. The second category of the kernel based methods is the Mean Shift Method. The Mean Shift algorithm aims to detect the region of a frame that is most similar to a reference model. For modeling either the reference object or the "key" object, probability density functions are utilized as well as color histograms. Support Vector Machines (SVM), the third category of kernel-based approaches, is a wide used classification scheme. According to these algorithms, each sample (usually pixel groups) of a video frame are classified as either "tracking object" or "non-tracking object". Such approaches can handle partial occlusion of the tracking object but they require a training phase. Finally, according to the Layering based tracking, each frame is separated to three layers; namely, shape representation (ellipse), motion (such as translation and rotation,) and layer appearance (based on intensity). Such approaches can handle tracking of multiple objects.

Concluding with the object tracking algorithms, we discuss about the Silhouette

Based Tracking approaches. These algorithms are used to track objects with complex shapes, such as fingers. Silhouette based methods utilize accurate shape descriptions for the objects. Silhouette based tracking approaches are categorized as either contour tracking methods, where a contour reshapes from frame to frame aiming to keep track with the object or Shape Matching algorithms, where only one frame is examined from time to time (without knowledge passed from the previous frame), using density functions, silhouette boundary and object edges.

2.5 Quality enhancement algorithms

The knowledge extraction algorithms discussed in the previous section use as input either frames or video streams. Such input is required to either enhance the quality of the modalities or to provide an initial layer of information for the next processing level. In this section, we will discuss some of the most important quality enhancement methods as well as the most common preprocessing algorithms.

2.5.1 Foreground/background identification

Foreground/background modeling identification is the process where each pixel of a scene is classified in two classes; either F (denoting the foreground) or B (denoting Background), which can be eliminated to a one-class classification problem, if uniform foreground distribution is assumed, as the intensity of a foreground pixel can randomly take any value (unless specific information about the foreground is available) (Elgammal, 2014). Foreground includes the surveillance subject while the background includes the rest of the scene. There are several approaches which can model the background, as depicted in Fig. 2·5. According to Single Gaussian background models, the noise distribution at a given pixel can be modeled as a zero mean Gaussian distribution. Thus, the intensity (or any other pixel feature) at a pixel is a random variable with a Gaussian distribution, which was widely reported in liter-

ature in the 90's. In case of colorful images, a multivariate Gaussian model is used. This model can be adaptive to slow changes in the scene (e.g., dust) by recursively updating the mean with each new frame. Single Gaussian Background models fail to model (usually) outdoor environments, where background is not static (e.g. leaves of a tree). In order to model such scenes, a generalization based on a Mixture of Gaussians has been proposed in the literature.

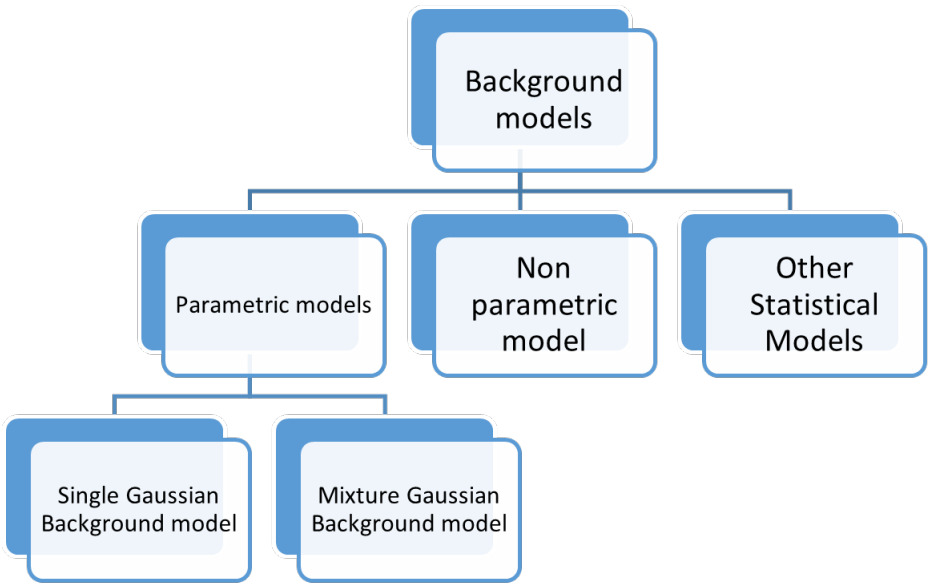

Figure 2·5: Background modeling approaches.

The need of modeling highly dynamic scenes requires a much more flexible background modelling. This leaded to the use of non-parametric density estimator for background modeling. All non-parametric density estimation methods (e.g. histograms) are asymptotically kernel methods, and a wide used non parametric model is the kernel density estimation technique, which estimates the underlying density and is quite general.

Lastly, in the literature there have been proposed other statistical techniques for background modelling. For example, linear prediction use the Wiener filter to predict

pixel intensity given a recent history of pixel values while linear prediction employs the Kalman Filter. Another approach has used Hidden Markov Models to model a wide range of variations in the pixel intensity. These variations are modeled as discrete states corresponding to modes of the environment, for example cloudy vs. sunny. Other approaches utilize background subtraction techniques which deal with quasi-moving background, e.g. scenes with dynamic textures. One robust algorithm of this approach is the *Auto Regressive Moving Average model*, where a Kalman filter was used in order to update the model.

Finally, a biologically-inspired non-parametric background subtraction approach has been proposed, where the pixel process is modeled as an artificial neural network.

As far as the features that are used for Background Modeling concern, intensity has been the most commonly-used feature. Alternatively, many research works use edge features. The use of edge features for background modelling is inspired by the need to have a brightness invariant representation of the scene. Another feature is the optical flow, which was used to capture background dynamics. Apart from pixel-based approaches, block-based approaches have also been used for background modeling. For example, block matching has been used for detection of changes between successive frames.

2.5.2 Image/Video quality enhancement algorithms

Image/Video enhancement algorithms are mandatory for any surveillance system. Low quality sensors and multivariate environmental conditions (e.g. fog, rain, extreme sunshine etc.) produce highly noisy video streams. Hence, enhancement algorithms are crucial for the robust function of applications such as object detection and object tracking. There are two main techniques for image enhancement depending on the domain each technique works; namely spatial based and frequency-based domain. Spatial based domain refers to the image plane itself, and algorithms in this class

process the image pixels directly while frequency-based domain processing techniques represent the image in the spatial domain and manipulate the spatial frequency spectrum. Research community has proposed several methodologies for improving the quality of image/video input, which can be categorized as shown in Fig. 2·6.

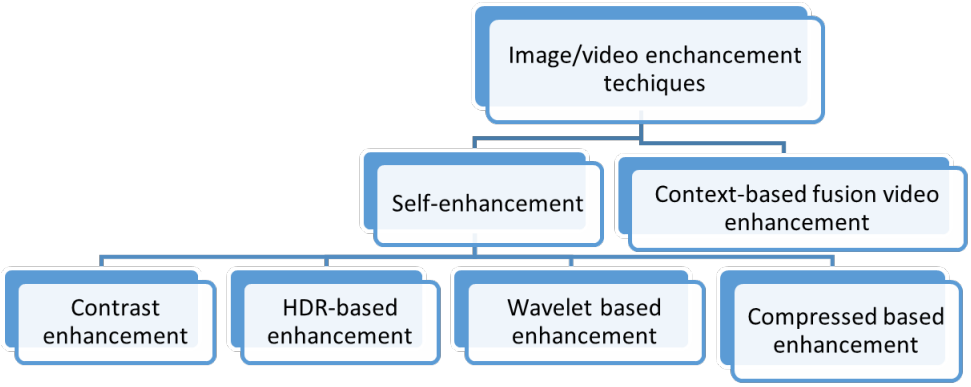

Figure 2·6: Categories of Image/video enhancement techniques.

Self-enhancement techniques refer to the techniques that use as input only the image/video under study. There are four categories in this class. The first category refers to modifications on the contrast map of an image. The aim is to adjust the local contrast in different regions of the image so that the "hidden" details in shady or bright regions are revealed. There are numerous algorithms for contrast enhancement which all aim at taking advantage the parts of the dynamic range that are "inactive". Widely used algorithms are power low rule, gamma manipulation, histogram equalization and tone mapping. Histogram equalization aims to uniformly distribute an image's histogram utilizing density functions. On the other hand, tone mapping techniques take under consideration the display device of video, trying to map the tone between the video input and the tone of the display device. HDR-based enhancement techniques are the second category of self-enhancement methods. HDR is a set of methodologies that offer a larger dynamic range of brightness between the brightest and the darkest pixel. HDR images can be produced by either combing multiple

images of the same scene taking under different exposure values or by using image processing algorithms. The third category utilizes wavelet transformation, producing a wavelet image, suitable for processing the image/video. The wavelet techniques utilize wavelet coefficients, wavelet shrinkage denoising or the dual-tree complex wavelet transform. Finally, the compressed based enhancement algorithms operate directly on the transform coefficients (e.g. Discrete Cosine Transform) of the images that are compressed. As far as the context – based fusion enhancement techniques concern, they utilize information from other modalities, or even from past data of the same sensing device in order to overcome poor light conditions and other environmental noisy situations.

2.5.3 Limitations

All of the aforementioned algorithms and techniques are innovative and provide solutions to by any means non-trivial problems. Yet, almost all of the approaches share, more or less, the same weaknesses. First of all, while the majority of video processing algorithms (such as motion detection) work fairly well, when we move to video analysis algorithms (such as human running detection), the response time of the systems increase and the accuracy decreases. Additionally, as debated in (Porikli et al., 2013), most of the test databases used to evaluate the performance of surveillance systems don't include heterogeneous datasets. Thus, the accuracy of proposed algorithms differs, sometimes to a great extent, when they are tested to real life scenarios, where the lighting and weather conditions constantly change.

Taking under consideration that nowadays the majority of the installed surveillance systems are CCTV based, there is a great need of addressing issues like robustness to environmental conditions, practical or even automatic effective calibration procedures (applicable to systems of hundreds and even thousands of cameras), dealing with crowded conditions and being able to handle pan-tilt zoom cameras. Addi-

tionally, most surveillance systems face technical limitations. The most usual ones are improper viewing angle, blind spots in coverage area, improper lighting conditions, improper recording resolution settings and too few cameras with too wide field-of-views. Yet, the proposed techniques (along with the required infrastructures) which are proposed (both from research and industry arena) address these limitations entail costs that can be prohibitive in many applications, such as warehouse monitoring and shop-lifting alerts.

While CCTV systems handle the recorded video stream in house, providing a level of privacy and security on the content, the scene is completely different when it comes to surveillance systems with IP cameras supported by cloud services. Streaming video content over Internet raises security and privacy issues which are difficult to tackle with existing technologies like VPNs or cryptography (Costa et al., 2017). The majority of the proposed systems do not deal with these issues which are crucial for a surveillance system, especially if the captured video streams can be viewed as potential law evidence to a court. Thus, there is a great need of designing approaches which will be more robust, more reliable and more secure, increasing the applicability and therefor the economy scale of surveillance systems.

2.6 Computing infrastructures

2.6.1 Cloud-based accelerators

Real time or near-real time response is perhaps the most important factor when it comes to surveillance systems. Automatic alerting upon a specific event is only valuable when it occurs within a time window after the actual event. Nowadays, surveillance systems, which meet the aforementioned requirement have been designed and deployed all around the world. Yet, the nature of the events which are recognized automatically from the systems are rather trivial, including object moving,

fire existence or object recognition. Nonetheless, surveillance systems face today a set of challenges, which involve car accidents detection, terrorist activities prediction or multipurpose behavioral analysis. These events require substantial larger computational resources, as they comprise complex calculations and non-linear models. On top of this, modern video sensors are able to capture HD and HDR footages, which facilitate the event detection algorithms and tackles, to a certain point, bad lighting conditions and other artifacts. The result of incorporating such sensors into surveillance systems is the proliferation of the produced data rates and of course the increment of the required storage size.

Both requirements for additional computational capabilities and storage size increment could be addressed by integrating surveillance systems with cloud infrastructures. There are not many reported surveillance systems in the literature, which use cloud services, either as SaaS (Software as a Service), as PaaS (Platform as a Service) or as IaaS (Infrastructure as a service).

One of these works is reported in (Rodríguez-Silva et al., 2012), where the proposed cloud infrastructure is used as SaaS and focus mainly on storage issues, using Amazon S3 platform. On the same track, (Li et al., 2011) describe a surveillance system for urban traffic systems, which is able to process massive floating car data coming from city taxis. Bigtable and MapReduce are explored as cloud technologies for not only storage purposes but also for computational processes. Finally, a resource allocation scheme for service management in cloud-based surveillance systems is described in (Hossain et al., 2012), where VM (Virtual Machines) resources are tuned based on QoS requirements, as depicted in Fig. 2·7.

2.6.2 Fog/Edge based accelerators

As discussed in the section 6.1, the concept of introducing cloud infrastructures and services into surveillance systems resolves (partially or fully) major limitations of

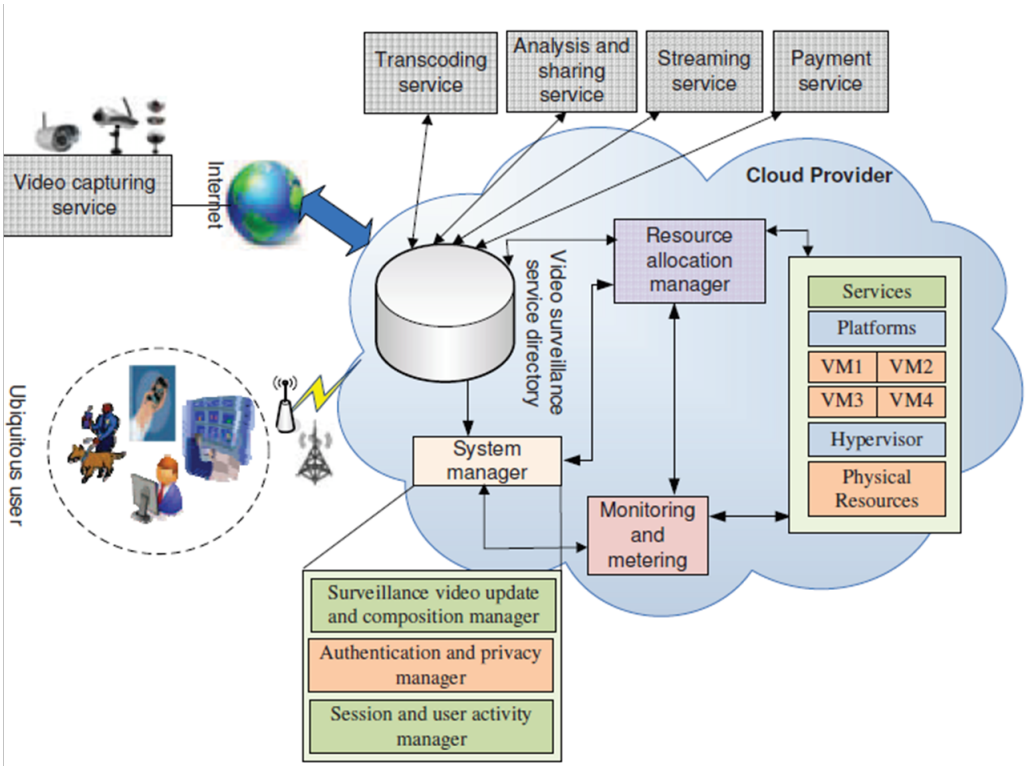

Figure 2·7: Proposed conceptual cloud architecture. (Hossain et al., 2012)

current systems, such as lack of computational power and restrictions in storage capacity. Yet, this approach introduces some new challenges that need to be addressed by a surveillance system that will be capable of meeting the requirements of the end users. More specifically, a surveillance system that utilizes cloud infrastructures needs to take into account the latency and the extra communication cost that is introduced between the sensors and the cloud infrastructure. Sending video streams to the cloud is by no means cost effective, especially if the video sensor has large resolution (e.g. HD video), while at the same time bearing in mind the *"best – effort"* characteristic of IP networks, the latency that is introduced is not only large but also fluctuating. These network characteristics prevent a cloud surveillance system from "reacting" to (near) real time events, such as car accidents.

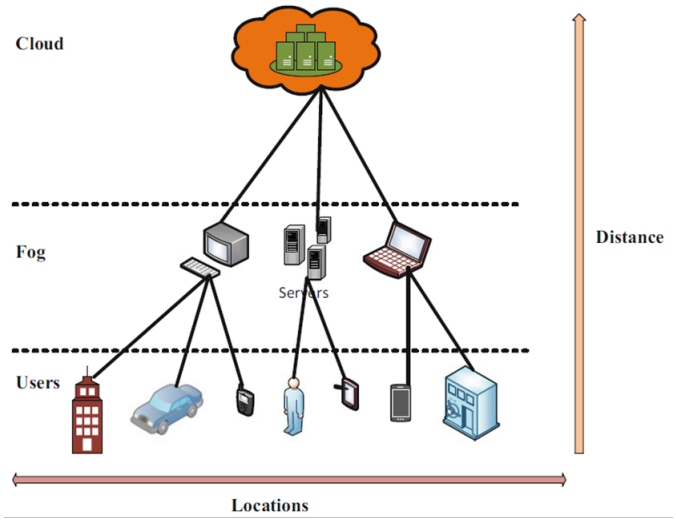

Figure 2·8: The Fog Computing conceptual architecture. (Chen et al., 2017)

A solution to these issues could be Fog Computing. Fog Computing (Fig. 2·8) is a paradigm that extends Cloud computing and services to the edge of the network. Thus, Fog Computing should not considered as a competitor to Cloud Computing but a complement technology that improves the characteristics of Cloud services. One

can describe Edge/Fog Computing as a distributed computation concept which is installed near (in terms of communication latency) the production of raw data. Fog Computing comprises many (and sometimes heterogeneous) devices that are capable of communicating and reallocating computational tasks. These devices have enough power to perform non trivial computational tasks, but in no case they match the capacity of a cloud system. The main advantage of Edge/Fog Computing is that if can offer to a surveillance system a first level of analytics extraction and decision making with minimum network traffic overhead and latency.

The research community during the last few years tries to utilize the concept of Edge/Fog Computing to surveillance systems, where video streams from Google Glasses© where captured and processed by either Google Glass device or the user's mobile devices (e.g. smartphone), depending on the battery life of the devices and the required computational power. The architecture is tested on several video processing tasks, such as face recognition and Optical Character Recognition (OCR). An urban traffic management and car accident system is described and tested in (Chen et al., 2017), where Fog Computing enables the near-real time vehicle tracking and its speed calculation. Following and projecting the logic of the last two sections, edge computing is introduced to surveillance systems. As defined in (Shi et al., 2016), Edge Computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services. Within the context of surveillance systems, edge computing refers to transferring computational and storage capacities from datacenters to the video sensors (or any other kind of utilized sensors), minimizing further (comparing with Fog Computing) and eliminating network latency. The paradigm of Edge Computing deployed to a surveillance system will require the usage of special hardware and / or software aside each video sensor. This hardware/software will be

able to perform a first level of video processing which can boost the performance of the surveillance system, in terms of response time and communication costs. For example, the edge systems could calculate the features (e.g. HOG descriptors) from the captured video stream and forward them to the next tier, decreasing the network requirements, as performed in (Zhang et al., 2016). In another research study, body-worn cameras are suggested to be used by police officers, aiming to provide specific analytics for law enforcement (Corso et al., 2016), while in (Shi et al., 2016) case-studies for cloud offloading and video analytics at the edge of a surveillance system are explored (Fig. 2·9).

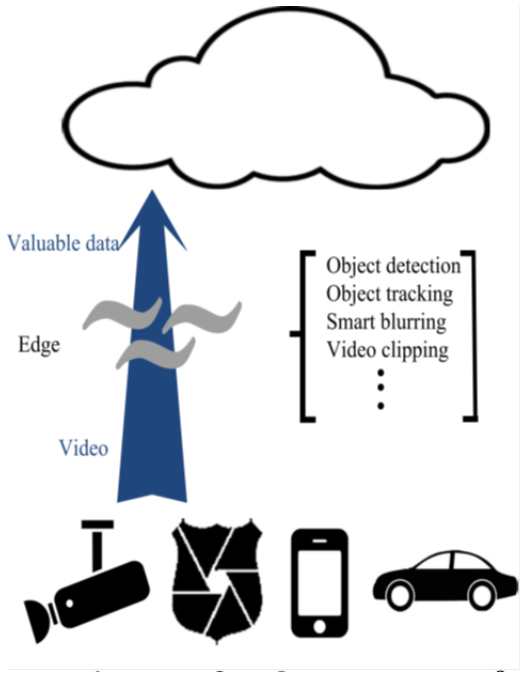

Figure 2·9: Overview of Edge Computing concept. (Zhang et al., 2016)

Taking under consideration the described Cloud, Fog and Edge Computing concepts within the context of surveillance systems, a promising approach is a "blended" architecture (Fig. 2·10), where certain characteristics (e.g. low-cost and proximity of the Edge layer, connectivity and power sustainability of the Fog layer and computa-

tional power and storage of the Cloud layer) of each approach are utilized, in order to maximize the efficiency of the system.

2.6.3 Deep learning methodologies

Deep learning approaches are intensively utilized during the last decade for addressing some of the most challenging problems regarding visual content, such as image classification, knowledge extraction and object identification. While the concept of deep learning regarding visual context was initially introduced through many years before, inspired by a biological model of the cat's visual system, it only produced tangible implementations at the end of the previous decade. One can mention three key developments. Namely, the high increase in the computational power and in the capacities of the processing hardware, the exponential decrease of the hardware's cost and the substantial advances in the machine learning algorithms.

Deep learning algorithms are a subclass of machine learning algorithms, which have the capacity of discovering multiple levels of distributed representations. The key word is "discovering", which implies that deep learning algorithms can identify the most important features that should be used for performing an information representation, such as object identification or human pose estimation. In order to achieve this, deep-learning approaches usually requires a (very) large dataset of annotated data. The features that deep learning approaches retrieve usually have a very important characteristic, when it comes to visual content analysis. They are invariant to irrelevant variations of the input. While for humans this task is trivial (e.g., identifying a lion regardless its pose), for many image processing algorithms, a change of an object's pose can alter the labeling output.

While the research community has proposed many algorithms, techniques and methodologies for deep learning algorithms, one can categorize the deep learning approaches in four classes (Guo et al., 2016). Namely,

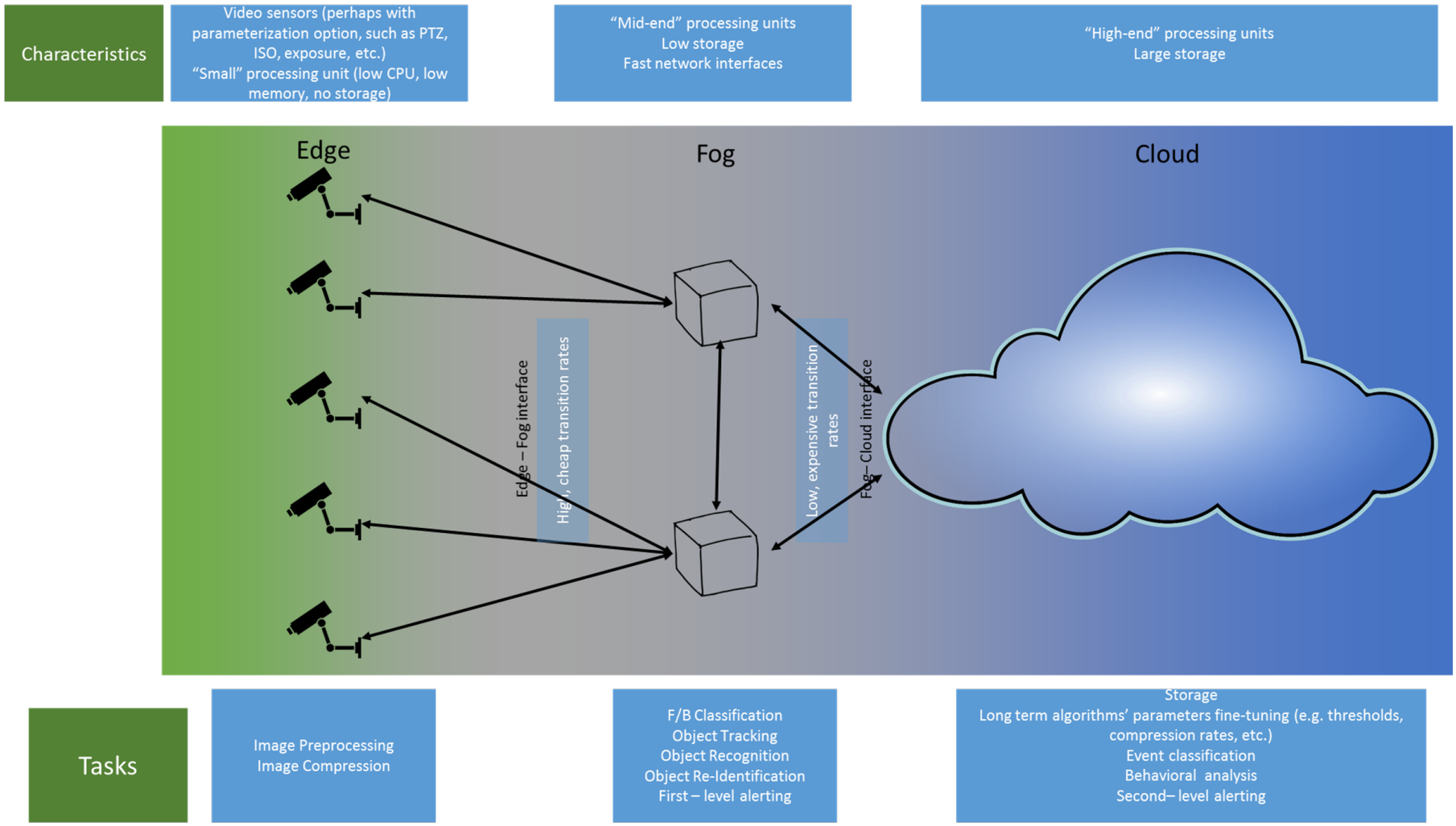

Figure 2·10: Cloud, Fog & Edge blended conceptual architecture.

i Convolutional Neural Networks (CNNs),

ii Restricted Bolzmann Machines (RBMs),

iii Autoencoder-based methods and

iv Sparse Coding – based methods

CNNs is probably the most common deep learning approach in the visual context. CNNs utilize multiple layers, which are trained in a robust manner. Effectiveness and robustness of CNNs have been proved by many research works in various computer vision applications. Some of the most important works in this directions are AlexNet ©, Clarifai©, VGG©, GoogLeNet© and SPP©. Restricted Boltzmann Machines (RBMs) were originally introduced by (Hinton et al., 1986). An RBM, is a generative stochastic neural network, (modified Boltzmann Machine), with the constraint that the visible units and hidden units must form a bipartite graph. This constraint allows for more effective training algorithms, such as the gradient-based contrastive divergence algorithm. Some of the most representative research works, which utilize RBMs are Deep Belief Networks, Deep Boltzmann Machines and Deep Energy Models.

Auto encoders is a special class of artificial neural networks, which utilize an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the input values. Auto encoders are trained to reconstruct their own inputs, which are then used for the training phase (this explains the auto in their name). In other words, they learn an approximation of a function, so as to produce an output that is similar to the input data. This results to the output vectors having the same dimensions as the input vectors. The encoder brings the data from a high dimensional input to a *'bottleneck layer'*, where the number of neurons is smaller than the input and output layers. Then, the decoder takes this encoded input and converts it back to the original input image. The latent space is the space in which the data

lies in the bottleneck layer. The latent space contains a compressed representation of the image, which is the only information the decoder is allowed to use to try to reconstruct the input as faithfully as possible. To perform well, the network has to learn to extract the most relevant features in the bottleneck. Some of the most important applications that utilized auto encoders are Sparse Autoencoder, Denoising Autoencoder and Contractive Autoencoder.

Sparse coding aims to learn an over-complete set of basic functions in order to describe the input data. Two of the most important advantages of the sparse coding are

(i) it can reconstruct the descriptors by using multiple bases and capturing the correlations between similar descriptors which share bases and

(ii) the sparsity allows the representation to capture salient properties of images and videos.

The most important research works that utilize sparse coding are Sparse Coding SPM, Laplacian Sparse Coding, Local Coordinate Coding and Super-Vector Coding.

2.7 Edge Computing at scale

During the last decade, academia and industry have introduced a set of architectures which common characteristic is the utilization of low-end hardware new the place data are generated. Such architectures, like Multi-access Edge Computing, Fog Computing, Cloudlet Computing, and Mobile Cloud Computing, while differ in several aspects, like communication network ownership, device mobility and device power supply, all share in common the same need for distributed architectures and approaches, which can enable the hosting of more demanding services (Carvalho et al., 2021).

Several research studies have been focused on enabling edge computing to support demanding latency sensitive applications. Author in (Kitchin, 2014) has proposed a scheme for handling mass video data coming from city surveillance services on heterogeneous digital devices. Zhou *et al.* (Zhou et al., 2015) have described a model for offloading cloud by utilizing an edge meshed network. Li *et al.* (Li et al., 2017) have proposed a general virtualization architecture, based on VMs, mainly focusing on its networking aspects. Chen *et al.* (Chen et al., 2016) have described an architecture which explores fog computing as a processing infrastructure for supporting dynamic urban surveillance streams. Dautov *et al.* (Dautov et al., 2018) have performed a comparison study among cloud, fog and edge computing for supporting intelligent surveillance applications.

The authors in (Satyanarayanan, 2019) provide a survey of the applications that can be supported from Edge Computing. Finally, the author in (Chen and Chen, 2018) provides a holistic vision about surveillance applications on edge/fog computing paradigms, where the basic concepts, challenges and opportunities are discussed. The proposed model has compared with the current state-of-the art literature and does not require any special virtualization (e.g., Virtual Machines) (Li et al., 2017) or distribution (Lee et al., 2017) (e.g., Apache Spark ©) middle-ware in order to perform the real-time calculation of AI analytics, offloading the edge devices from the substantial overhead these approaches require. Additionally, surveillance applications are decomposed in virtual functions that are deployed in nodes with the available resources. Such functions are scaled up based on demands.

Deep learning techniques have been widely used during the last years to extract information from various kinds of data (Khan, 2018). Depending on the characteristics of input data, several architectures for deep learning have been proposed, such as the recurrent neural networks (Ranzato, 2014), convolutional neural networks (Rawat

and Wang, 2017), and deep neural networks (Sharma and Singh, 2017). As deep and convolutional networks do not have the capacity to manage temporal information of input data, areas involving data such as text, audio or video, recurrent neural networks (RNNs) are usually applied. More specifically, there are two types of RNNs: discrete-time RNNs and continuous-time RNNs (Gallagher et al., 2005). The main characteristic of the RNN architecture is a cyclic connection, which enables the RNN to possess the capacity to update the current state based on past states and current input data.

Long short-term memory (LSTM) networks have been proposed for input data which hold dependencies with a large temporal distance (Neil et al., 2016), which fit the problem described in the present model.

2.8 Edge Environment simulators

The technological developments of the last decade in the area of networking and embedded systems have resulted to an exponential increase of micro-processing units, which can be deployed near the production of primal data. These units formulated the paradigm of Edge Computing. Edge Computing is an umbrella term which describes an extensive range of computing concepts, such as mobile computing, multi-access edge computing (MEC) (Giust et al., 2017), open edge computing (Dinh et al., 2013) and fog computing (Kim, 2016). The underline denominator of the aforementioned approaches is the effort to minimize the network distance and thus increase proximity between data (usually generated by sensing devices and communicated through streaming protocols) and processing devices. Time sensitive services benefit from the edge computing model, as the round-trip network delay, implied by the cloud computing based services, is omitted. Additionally, services which can not access cloud infrastructures can explore Edge computing for deployment.

Furthermore, the network infrastructure of the edge computing provides a lower transmission delay than the cloud computing because the clients do not encounter the wide area network (WAN) delay. Thus, services like video analytics or biosignals processing can be implemented more efficiently on the edge of the network (Orsini et al., 2018).

Both academia and industry are showing an increasing interest towards edge computing and many researchers and engineers are designing new models and novel approaches. Such approaches though pose a challenge from the design point of view, as a wide set of parameters and performance criteria need to be considered. During the design phase, there are three options to be explored: (i) cloud environments (Pacheco and Hariri, 2018), (ii) experimental test beds (García-Pérez and Merino, 2018) and (iii) simulators (Yaseen et al., 2018), in order to evaluate the proposed schemes. For each option, certain advantages and disadvantages can be discussed. An actual cloud environment is usually costly and requires special virtualization frameworks for deployment. Likewise, designing on experimental test beds brings in difficulties regarding the expandability of the experiments and scalability of the proposed architectures (D'Angelo et al., 2017). On top of that, as the main category of edge processing units are mobile systems, emulating their software environment is a challenging task.

Edge computing, fog computing, open edge computing, and MEC, while presenting some dissimilarities, can be considered as similar conceptions. In the present study we use the term edge computing, for consistency.

A useful review on the evolution of the edge computing paradigm is provided by Taleb *et al.* (Taleb et al., 2017). (Satyanarayanan et al., 2009) introduced the Cloudlet paradigm. Cloudlets can be thought as micro-clouds close to mobile users, which can deploy and manage their own virtualized environments. About the same period, Cisco© proposed fog computing (Kim, 2016), a concept comparable to the

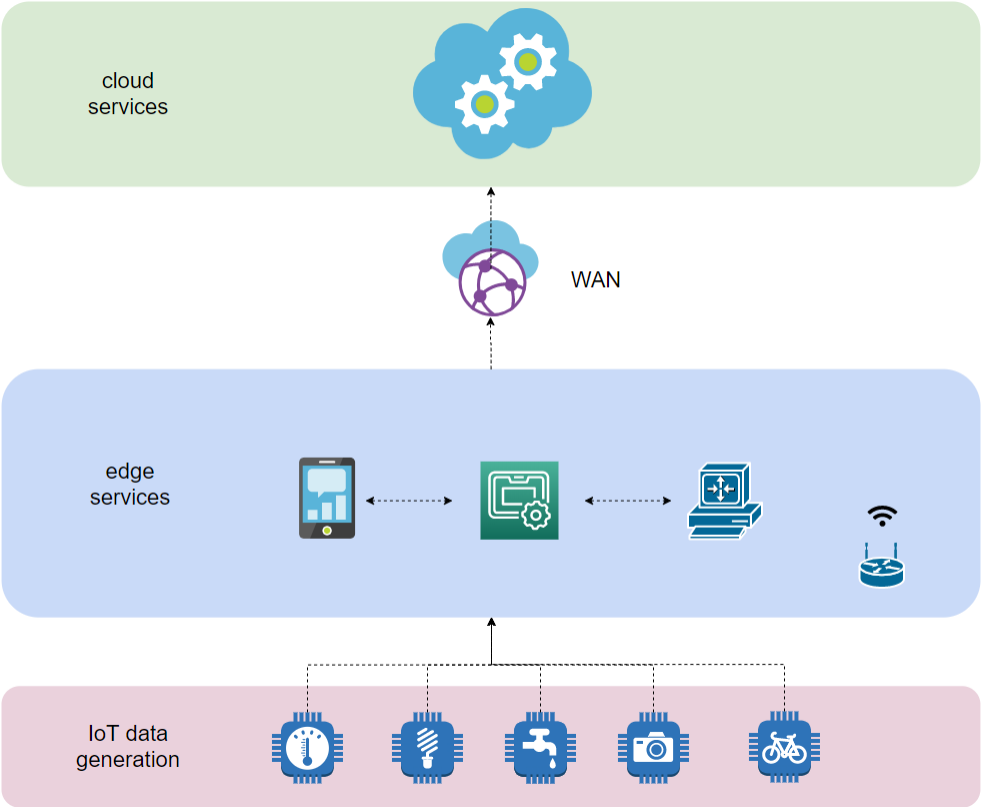

Figure 2·11: Cloud - Edge - IoT conceptual architecture.

main ideas as cloudlets. Fog computing architectures most usually comprise devices placed in the edge of the network, with the capacity of of wireless communication.

The aforementioned approaches draw the attention of the telecommunication service providers and public institutions. Thus, European Telecommunications Standards Institute (ETSI) introduced the mobile edge computing (MEC) concept, aiming to adjust the edge computing paradigm in mobile cellular networks (Farris et al., 2018). The objective of this concept is to enable the support of real time access to high-end services by adjusting the edge and cloud computing capabilities into the edge of the radio access network (RAN). To expand the MEC paradigm from cellular networks to alternative wireless access technologies (e.g. WiFi), ETSI modified the name of the concept to multiaccess edge computing (keeping though the abbreviation the same).

A typical edge computing architecture and end user devices are depicted in Fig. 2·11. The proliferation of cloud based services resulted to numerous cloud computing simulators (Byrne et al., 2017). Basically, most of these simulators provide the computational models for the virtualized cloud environments, which can simulated the basic aspects of the virtual nodes, such as CPU, RAM memory, storage and energy consumption. The models incorporated in the simulators usually consider at least three of the aforementioned characteristics.

Three of the most well established cloud simulators are GreenCloud (Kliazovich et al., 2012), iCanCloud (Núñez et al., 2012) and CloudSim (Calheiros et al., 2011). GreenCloud (an add-on of the well-known NS2 simulator) is a framework which has advanced models for energy usage and consumption for both communication and computational tasks. GreenCloud can utilize the full TCP/IP stack from the NS2 library, enabling the detailed modeling of the energy consumption on the network elements. Nonetheless, NS2 backend requires substantial CPU and memory capac-

Table 2.2: Comparison of the well-established simulators.

		Resource modelling	Dynamic simulation	Virtualization modelling
Cloud	**GreenCloud**	-	-	✓
	iCanCloud	✓	-	-
	CloudSim	✓	✓	-
Edge	**IoTSim**	✓	-	✓
	SimIoT	-	✓	-
	iFogSim	✓	-	✓

ity, affecting the simulation time of the produced scenarios. Built on OMNeT++, iCanCloud is another well-established cloud simulator. The comparative advantage of iCanCloud is the capacity to simulate large-scale environments with thousands of nodes, by supporting great extensibility, scalability and performance indicators. iCanCloud provides a graphical user interface to describe the simulation scenario.

When it comes to simulate Infrastructure-as-a-Service cloud computing environments, CloudSim is probably the optimal option, as it is designed for modeling both cloud components such as datacenters, virtual machines and hosts, and source provisioning policies such as CPU, RAM memory, storage, and network communication models (Makaratzis et al., 2018).

Edge environments, compared with the cloud setups, share diverse characteristics, as far as devices, virtualization environments, networking, user access and service deployment. For this, additional aspects need to be integrated in the simulation frameworks which plan to support edge computing environments. These aspects can be summarized in three categories:

1. Edge node mobility profiles, as mobility is usually not considered in cloud environments,

2. Data generation devices (e.g., sensors) integration and

3. Reliability profiles for the edge devices.

For supporting these capabilities, simulators specialized for edge environments have been introduced. Among the most popular ones are SimIoT (Sotiriadis et al., 2014), IOTSim (Zeng et al., 2016), and iFogSim (Zeng et al., 2016). IOTSim is simulator which is built upon the engine of CloudSim. It is proposed for simulations which require large volumes of data to be sent out to cloud infrastructures. SimIoT is developed by extending the SimIC (Sotiriadis et al., 2013) simulator and, in principle, it integrates an IoT layer to the SimIC, allowing edge devices to request and access cloud resources. Finally, iFogSim runs on top of CloudSim and it has been developed for simulating IoT and fog environments by modeling components like the sensors, actuators, fog devices and the cloud infrastructures. iFogSim supports the establishment service access from edge devices from fog servers. Table 2.2 summarizes the reviewed simulators, in terms of their capacity to model resource allocation, virtualization technologies and dynamic changes during the execution of a simulation.

The main limitation that the aforementioned simulators share is that the simulation scenario is static and can not be updated during a runtime. This acts a bottleneck for simulating VFC model, where runtime decisions about VF placement or migrations (see Chapter 5) are required.

2.9 Migration on the network Edge

A migration process of an edge service (or even a virtual environment) from an edge device to another can be either stateless or stateful. When referring to stateless (Meng and Lu, 2021), the migration usually comes down to a deployment (or a redeployment) of a stateless virtual function on another edge device. Stateless migrations are deployed on situations when a device hosting a virtual function unexpectedly fails and all the runtime data of the virtual function are lost. In this case, migration models need to consider the selection of the most suitable node to undertake the

execution of the virtual function.

Yet, things are far more challenging when stateful migrations need to be performed (Jin et al., 2021). By stateful, we consider both runtime and persistent data, which need to be transmitted to the receiving node in order to restore the virtual function exactly in its state before the migration process. Yet, as discussed in the previous section, differently from the Cloud systems, persistent data (usually written on the secondary memory) are not utilized by edge virtual functions. Hence, most of the edge migration models usually consider runtime data from the source to the target node and then applies this state to a base image of a virtual function. This image may be acquired from shared repository, or transmitted through the edge orchestrator, or it could even be available on the receiving node thus making only the runtime state impact the overall migration time.

Additionally, when it comes to edge networks, both the low capacity (in terms of processing power) of the edge devices and the network interconnecting links force migration models to reduce both the computational and the network overheads. The non-trivial aspect of this statement is though that these two remarks often appear opposing characteristics, as improving one deteriorates the other.

2.10 Future trends in Edge Computing and Video Surveillance Systems

2.10.1 Surveillance Systems and augmented reality

Augmented Reality, in the context of surveillance systems, refer to the information depicted on the operator's screen(s) on top of the video stream captured by the surveillance cameras of the system. The type of the projected information range from static information to object tracking trajectories, dynamic labeling of detected objects and missing or hidden objects. Some of the most important studies on this

field come from surveillance system used for military purposes. For instance, in (Hall and Trivedi, 2002) a scheme is proposed for observing multiple video streams and a visualization system is proposed by merging dynamic imagery with geometry model of a battlefield visualization. In (Kumar et al., 2000), an augmented visualization of urban locations is reconstructed using offline video streams and 3D location models. Finally, a system which automatically detects humans and vehicles from multiple video streams and then extract and place selected frames on a map, thus reducing the workload of the operator, is described in several research studies.

On a similar context, (Adhikari et al., 2016), a surveillance and rescue system is described which automatically combine computer-generated imagery with real-world imagery in a portable electronic device by retrieving, manipulating, and sharing relevant stored videos. Proposing similar technologies, the work presented in (Sebe et al., 2003) describes a visualization system for video surveillance based on an Augmented Virtual Environment (AVE) that fuses dynamic imagery with 360 and 3D models in a real-time display to assist observers and users to easily and effectively comprehend multiple video streams of temporal data and imagery from random views of the scene, where moving objects are detected and tracked in video streams and visualized as 3D elements in the AVE scene display in real-time. Finally, numerous research works have been presented where augmented reality is used to support operators watching video streams from surveillance cameras by offering functionalities like removing immobile items (over certain time frame) from a scene, providing text-based and sound-based messages or even proposing areas in the scene that the operator should pay attention to.

2.10.2 Challenges

During the past three decades, an enormous set of works addressing the problem of automatic (or semi-automatic) surveillance has been proposed by the research commu-

nity. Main subtasks that were studied were object tracking, object re-identification, object recognition and image enhancement. Within this framework, many excellent studies have proposed algorithms and systems which address the aforementioned problems with (more than) acceptable accuracy and robustness. Yet, a lot of work still needs to be done. Most of the video surveillance systems seem to share two common limitations. The first limitation refers to a (too) high false alarm rate in detection of interesting events within the surveillance scene. This drawback causes various problems to the owners of the surveillance systems and they usually decide to deactivate automatic alerting features. Secondly, existing surveillance systems fail to function properly under all real-world conditions, such as rain, fog, snow, blowing dust, water on the lens or image plane artifacts.

In order to overcome the aforementioned limitations, new algorithms and techniques need to be developed, increasing the accuracy and the robustness of the surveillance systems. Besides addressing flaws of already established surveillance systems, researchers working on video analytics should bring surveillance to the next level, working on the following topics (Fig. 2·12).

- **Cloud/Fog/Edge infrastructures integration**. Cloud technology seems to match perfectly with surveillance systems, as it can offer both the missing computational power video analytics require and the storage capacity usually a surveillance system needs. Cloud infrastructures are expected to facilitate installation and management of surveillance systems, shifting the paradigm from standalone applications to Software-as-a-Service. This will allow surveillance systems to use different video analytics and alerting mechanisms when it is required and for the time period it is required. Bearing in mind the cost transmitting a video footage to a cloud system and the cost of cloud storage, new compression algorithms must be used, which will maintain the accuracy of the

Figure 2·12: Research trends of surveillance systems.

video analytics algorithms while reducing the aforementioned costs. Cloud technology, as argued in paragraphs 2.6.2 and 2.6.3, can be supported and extended by Fog computing and Edge computing. More specifically, Fog and Edge computing can address the delay overhead that cloud services usually impose to a surveillance system by transferring computational power closer to the source of the event. By calculating features and analytics close to the sensors decreases the required network bandwidth and increases the response time of the system. Thus, these approaches can be used to cutting-edge approaches like automatic drone navigation or automatic field of view rearranging.

- **Communication protocols between surveillance systems.** Despite the fact that surveillance systems become more and more popular, there is no specific protocol for communication among them. Such protocols would be extremely useful for public safety scenarios and terrorism prevention, facilitating

information exchange between different surveillance systems deployed around a city. Thus, analytics such as object re-identification and object tracking would be possible among different and heterogeneous surveillance systems.

- **Modality fusion**. Apart from video, which is the dominating sensing technology for surveillance systems, other modalities can facilitate monitoring and alerting tasks. Such modalities are audio, thermal video, night vision video, HDR video and GPS tags. Thus, algorithms and techniques are required, in order not only to seamlessly fuse these modalities to a single output but also to automatically decide which modalities are more suitable for different conditions or for different tasks. These approaches, among other applications, are expected to provide to autonomous vehicles (such as drones) the functionality of "deciding" which sensors are more appropriate to use on different situations.

- **Analytics and scene reasoning.** The ultimate aim of an intelligent surveillance system is to automatically produce high-level information of the recorded scene, such as objects identification and motion recognition. Other tasks, such as tracking of individual people in crowds, keeping track of moving objects that are temporally occluded, and tracking and understanding interactions between multiple targets are further challenges that aren't yet reliably addressed. While the research community has proposed an extended set of algorithms and techniques in this area, higher levels of accuracy and applicability are required.

- **Surveillance databases & event oriented query languages.** The usual scenario of a surveillance system is to store the video footage for a pre-defined time-frame in order to use it in case of a future events, related to the area under surveillance. In such scenarios, the common practice is to review the video streams which is a rather time-consuming and resource demanding task. As we

use surveillance systems to capture events, surveillance databases must be event oriented, improving not only the workflow of a person seeking a specific event, but also the storage capacity of a system, as we will avoid the pointless saving of the whole video footage and focus on storing the events. Such databases will be integrated with event oriented query languages, in order to facilitate seeking tasks and high level knowledge extraction tasks.

- **Augmented reality on surveillance systems.** Offering in real time (or in near-real time) information, analytics and metadata about a monitoring scene would undoubtedly help surveillance operators to work with several monitors and with crowded scenes. Thus, producing virtual reality information and over layering it with the actual video footage is a challenging task that needs to be further addressed. Additionally, generating an auditory display for complex scenes is very appealing to support situational awareness in surveillance systems. Approaches like these are expected to improve the workflow of monitoring.

- **Virtual reality**. As the number of the video sensors of a scene/area increases, the operator's monitoring work becomes non-trivial, as she/he has to constantly pay attention to multiple screens. As already argued, augmented reality can facilitate this workflow through adding an intelligent layer to the monitoring screens. Another approach to achieve this workflow facilitation is Virtual Reality (VR), where a set of algorithms and techniques will reconstruct a 3D (360°) world from the video sensors, in which the operator will be able to walk through and observe certain features, such as objects and individuals. While such solutions have been proposed in the literature (Du et al., 2016) a lot of work still needs to be done, improving the required algorithms for gaze direction computation, camera scheduling, collaborative tracking and Virtual Reality content streaming.

- **Deep learning algorithms.** While supervised deep learning algorithms have performed extremely well, when compared with other approaches, unsupervised learning is expected to play an important role in reviving interest in deep learning. Unsupervised learning is expected to produce representations and relationships which are not obvious even to animals and humans. Also, deep learning approaches are expected to mimic even further human vision, producing systems that are trained end-to end to decide where to in the field of view the system should focus.

- **Security at the edge of the network.** The very nature of an edge network creates a challenging environment for any well established security architecture which is applied in cloud environments. A recent model proposed for addressing security in the edge is Zero Trust (ZT) security model (Samaniego and Deters, 2018). According to ZT, entities are not considered trusted based on their network location (perimeter-based security architecture). Yet, all entities need to manifest a 'trust' token every time a request is address to it, without relying on implicit trust. While block-chain is already proposed as an implementation path for ZT architectures (Dhar and Bose, 2021), designing, implementing and testing new schemes for implementing ZT on the edge is still a great challenge to consider.

All of the aforementioned remarks share common ground with edge computing, and most of this ground has not yet been explored. More specifically, ad hoc protocols suitable for non-centralized services, like edge-based surveillance services are mandatory for integrating large-scale systems. On top of that, edge-based distribution schemes are necessary for supporting demanding computational tasks, like artificial intelligence analytics, deep learning models and virtual reality rendering. Such distribution schemes also require edge-based storage solutions and data redun-

dancy models. Finally, security and data integrity require innovative solutions when projected at the edge of the network.

2.11 Conclusion

After surveying the current status of surveillance systems, both from the algorithmic and the systemic / infrastructure point of view, several conclusions can be drawn regarding the available technology nowadays, the technological limitations and the future challenges of the area.

Video surveillance systems have been introduced almost fifty years ago, through CCTV systems, requiring a substantial number of manpower, analog to the number of the installed video sensors, leading to high operational costs. The majority of the research studies on surveillance systems the last five decades are trying, to substitute the operator with a video processing algorithms which will be able to perform certain tasks. While all of the effort put on this non-trivial mission has produced some really innovative and brilliant algorithms, these are only limited to a small set of tasks, like face recognition and object detection and tracking. The accuracy of these algorithms is usually far from satisfying when the scene conditions are not perfect.

Research developments during the last decade have revitalized the expectations for automatic surveillance systems. These developments mainly involve machine-learning, deep-learning algorithms and distributed computational infrastructures, like cloud, fog and edge computing. These methodologies, combined together, are expected to improve the accuracy of surveillance algorithms, proposed new smart analytics and reduce the response time of the systems, in order to produce meaningful alerts.

Limitations

As already discussed, there are a lot of research challenges that need to be addressed. Among these, special attention needs to be placed on optimization techniques that will automatically redistribute the computational power among edge, fog and cloud agents, based on specific performance, cost and privacy criteria. Such optimization techniques will boost the performance of the surveillance systems, enabling at the same time a new paradigm, Video Surveillance as a Service (VSaaA).

More specifically, the main deployment paradigm for deploying surveillance services nowadays is cloud computing. This paradigm implies the transmission of the collected video streams to a centralized hub, posing a high communication latency. Additionally, having video streams transmitted and stored to third-party servers raises privacy and security issues, which many users consider of paramount importance.

As already discussed, edge computing, by design, can address the two main limitations of cloud based surveillance systems. Yet, the low end devices which comprise the edge environments, along with high heterogeneity and mutable characteristics of both devices and network, deter the direct deployment of demanding streaming applications, like surveillance services.

These limitation aspects of edge environments drove the core inspiration for this PhD project. Within the next chapters, a model for enabling AI analytics and other heavy computational tasks to be deployed on an edge environment is proposed. Upon the basic model, auxiliary models for results caching And virtual function migration are described. Finally, evaluation of the proposed models, both on simulation environment and on a experimental setup provides proof-of-concept evidence that the proposed model is vital and can support actual use case scenarios.

Chapter 3

Proposed Virtual Function Chaining model

This chapter introduces a novel distributed model for handling in real-time, edge-based Artificial Intelligence analytics, such as the ones required for smart video surveillance. The novelty of the model relies on decoupling and distributing the services into several decomposed functions which are linked together, creating virtual function chains (VFC model). The model considers both computational and communication constraints. Theoretical, simulation and experimental results have shown that the VFC model can enable the support of heavy-load services to an edge environment while improving the footprint of the service compared to state-of-the art frameworks. In detail, results on the VFC model have shown that it can reduce the total edge cost, compared with a Monolithic and a Simple Frame Distribution models. For experimenting on a real-case scenario, a testbed edge environment has been developed, where the aforementioned models, as well as a general distribution framework (Apache Spark ©), have been deployed. A cloud service has also been considered. Experiments have shown that VFC can outperform all alternative approaches, by reducing operational cost and improving the QoS. Finally, a caching and a QoS monitoring service based on Long-Term-Short-Term models are introduced.

3.1 Introduction

Artificial Intelligence (AI), as expressed by the latest developments of Machine Learning (ML) and Deep Learning (DL), has produced numerous models which are mature enough to reach the market massively during the next few years. The main reasons for this, mainly involves the improvements on data-capturing devices, the re-engineering of several ML algorithms and the release of ML and DL toolkits, like PyTorch and TensorFlow (Jain et al., 2019).

Video analytics is an umbrella term for describing applications like object tracking, pedestrian detection, face recognition, behavioral analytics etc. The common business - technology model for deploying AI surveillance services nowadays is Cloud Computing (He et al., 2018), where the captured video streams are uploaded to a centralized data center. This imposes a round-trip time to the throughput of the service which, in many cases reduces significantly the Quality of Service (QoS). This leads the service providers to either reduce the processing frames per second (fps) or lower the resolution of the captured videos, nullifying the advances of new video sensors, like UHD and HDR sensors.

Edge Computing has been proposed as a computing paradigm according to which the data are processed *'near'* the generating data devices and comprises many low-capacity devices (Sun et al., 2019). While this paradigm addresses the large round-trip times of Cloud Computing, the QoS is now limited due to the capacity of the edge devices. Not only academia, but lately industry has placed focus on Edge Computing, by providing software (e.g., Google Lite TensorFlow (Demosthenous and Vassiliades, 2021)©) and hardware (e.g., NVIDIA Jetson AGX Xavier©(Hossain and Lee, 2019) and AWS DeepLens©(Khan et al., 2020)) solutions are suitable for edge processing. As discussed in several works (Alnoman et al., 2019), (Carvalho et al., 2021), current edge computing approaches face several challenges and limitations. These limitations

mainly involve the insufficient computational capacities for heavy loaded tasks, like AI model training and the low storage space usually the edge devices are equipped with.

This chapter proposes a novel distributing framework which explores the Virtual Function Chaining (VFC) concept inspired from the Software-Defined Networks and enables the real-time inference for surveillance applications at the Edge, supported by edge learning services build on deep learning models with the capacity to monitor, assess and predict the QoS of the supported services. In this model (Fig. 3·1), an AI smart video analytics service is decomposed to a set of Virtual Functions (VFs), which can be deployed on different edge devices. Using these VFs, a VFC is created which process the streaming data in a distributed fashion. For example, in Fig. 3·1, there are two $VFCs$ presented (purple and orange chains), each one of which implements a different service. VFO (Virtual Function Orchestrator) is responsible for deploying the $VFCs$, along with the auxiliary services discussed in the next chapters.

The VFC framework deploys several modules which aim to optimal design the service, monitor its QoS metrics and fine-tune its configuration in order to avoid failures. More specifically, a computational engine is responsible for proposing the optimal setup of the VFC while an edge-learning service monitors the performance of the edge devices and propose possible alterations.

The VFC architecture for effectively distributing an video analytics service to the Edge architecture seamlessly integrates $VFCs$. The proposed architecture's services are mainly hosted on the Virtual Function Orchestrator (VFO). Additionally, a model for designing the optimum setup for a VFC, in terms of VF instances and VF placement on the edge devices is presented. Each VF may appears in the VFC at several instances (replicas) and interconnected $VFCs$ (Fig. 3·2). The contributions of this chapter mainly include:

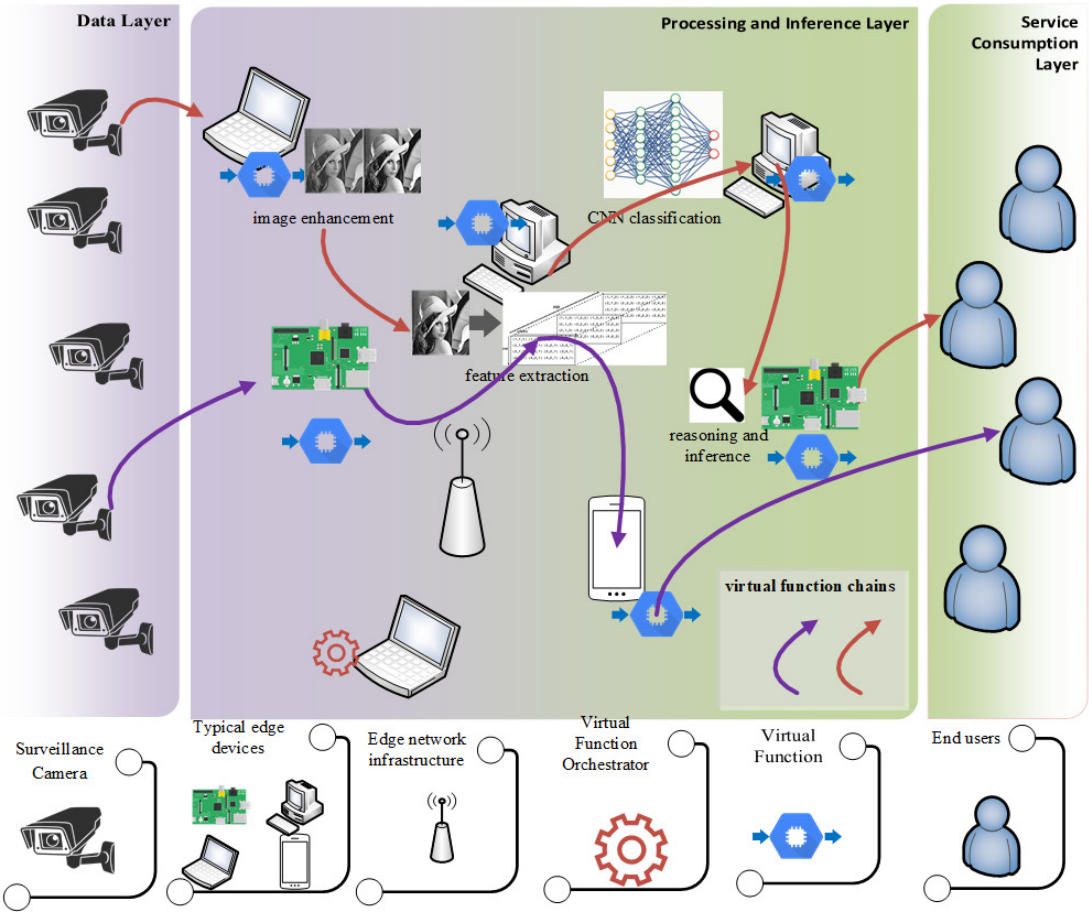

Figure 3·1: Conceptual architecture of the proposed system.

1. An edge learning service is built on deep learning models for assessing the QoS of the deployed services and alerting in case a VFC is about to fail.

2. A caching mechanism, which demonstrates the scalability of the proposed model when multiple services are requested.

3. A prototype of the described architecture, which is used to evaluate the described models and provide a proof of concept, in terms of effectiveness and feasibility, on enabling $VFCs$ as a model for real-time AI surveillance applications.

Additionally, within this work an extensive comparison with generic distribution engines (Apache Spark©) is presented, along with a set of experiments.

While the proposed model is inspired by the Service Function Chaining (SFC) concept, it alters and extends several of its features, in order to meet the requirements of video analytics. Service Function Chaining refers to the use of Software-Defined Networking (SDN) programmability to create a service chain of connected (virtual) network services. One advantage of using SFC is to automate the way virtual network connections can be set up to handle different types of traffic flows.

- First, it introduces a load-balancing mechanism in order to support the QoS, which monitors the output of the service and rearranges the VFC automatically.

- Additionally, it extends the SFC model by allowing one VF instance to be part in several $VFCs$ (e.g., face detection, gender identification, etc.), so that several video analytic services can be deployed simultaneously.

The different deployment modes of the $VFCs$ are presented in Fig. 3·2. Basic VFC deployment refers to the case according to each VF of a VFC is deployed to a different edge node, without any replications. VF replicas deployment describes the

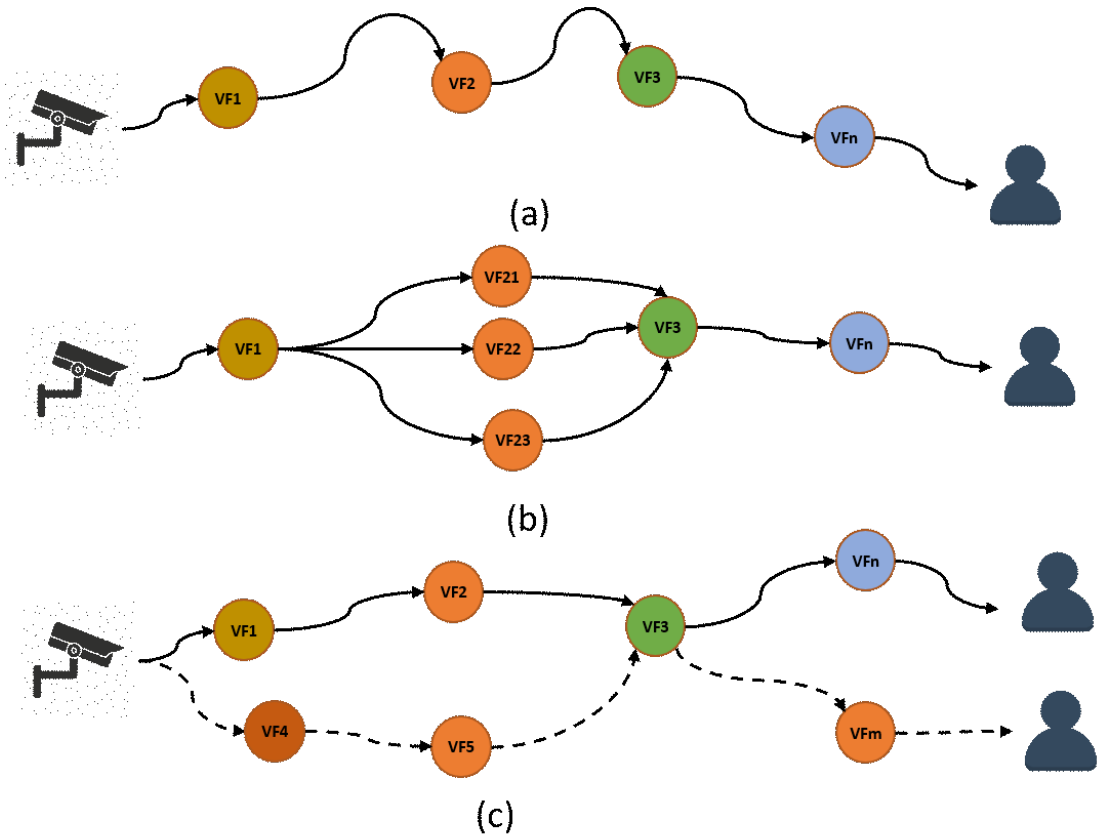

Figure 3·2: Different modes of VFC deployment. (a) Basic VFC deployment, (b) VFC deployment with VF replicas and (c) Two $VFCs$ with caching enabled.

case where one or more VF are deployed in replicas in order to divide the incoming streaming tasks and thus reduce the required processing time of this specific VF. Finally, the caching deployment presents the case according to which, one VF is part of two (or more) VFCs, enabling the utilization of cache data.

The management of the VFC (autoscaling, QoS monitoring, etc.) is facilitated by an edge-learning model with the capacity to assess the performance of the edge devices within a specific time-frame and inform the VFO about possible failures.

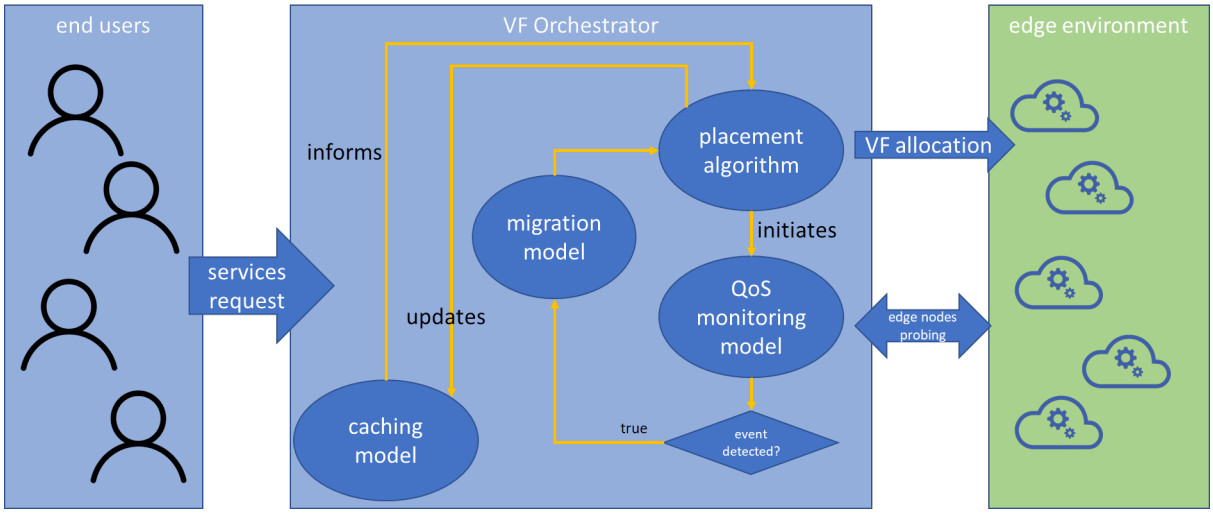

Figure 3·3: Logical architecture of the proposed VFC model.

More specifically, in Fig. 3·3, all the submodules of the proposed model are presented, along with their interconnections and dependencies. When a user requests from the orchestrator (VFO) a specific service, VFO executes the placement algorithm and assign each VF to the appropriate node. At the same time, the QoS monitoring model establish a mechanism for collecting data about the utilized edge nodes and as soon as a possible overload is detected, it triggers the migration model for assigning a VF to another edge node. At the same time, VFO checks with the cashing module whether there are other services up and running with a common VF. If so, the placement algorithm is informed and combines the appropriate VFs.

The rest of this chapter is organized as follows. Section 2.7 provides a brief state-of-the-art approaches for utilizing edge for hosting AI services, while Section 3.2 describes the VFC model. Section 3.3 includes the implementation details of all the tools developed to evaluate the proposed concept. In Section 3.4, the results from the experiments are presented. Finally, the results are discussed in Section 3.5.

3.2 Virtual Function Chaining Model

Aiming to enable edge as a real time inference mechanism for AI video analytics services, a generalization distribution framework is proposed, according to which a video analytic service is decomposed to a set of VFs, creating a VFC. The proposed model aims to facilitate the efficient offloading of surveillance cloud services to a cooperative distributed edge environment, where heterogeneous edge devices formulate a service chain and jointly implement a service.

The basic principles of the proposed model (Figure 3·1) are the following:

- Each surveillance service is decomposed to set of processes. Each process implements certain tasks, like image enhancement, edge detection, AI model inference, etc.,

- Each process instantiates a VF and is deployed as an edge node. Each VF comprises three parallel threads: the Input Queue, the Output Queue and the Running agent (Fig. 3·4). The running agent implements the logic part of the VF (e.g., image filtering) while the Input and Output queues handle the packaging and communication of the VF with the previous VF and the next VF in the VFC respectively. The communication between the VFs is unidirectional, as surveillance services are streaming processes. This means that when there is an information flow from VF_i to VF_j, there is no communication from VF_j to VF_i by default.

- A surveillance service is realized by a VFC, similar to the service function chaining proposed by the IETF WG (Halpern and Pignataro, 2015). A VFC must include at least one instance of each VF. The main concept of the model proposed by (Halpern and Pignataro, 2015), includes network services, like firewall and packet filtering.

- VFO manages the VFs allocation to the physical devices and established the communication channels among them. Additionally, VFO monitors the performance criteria of the service (e.g. processed frames/sec, total cost, etc.) and performs actions in order to meet them, while hosting the edge learning AI model for QoS monitoring.

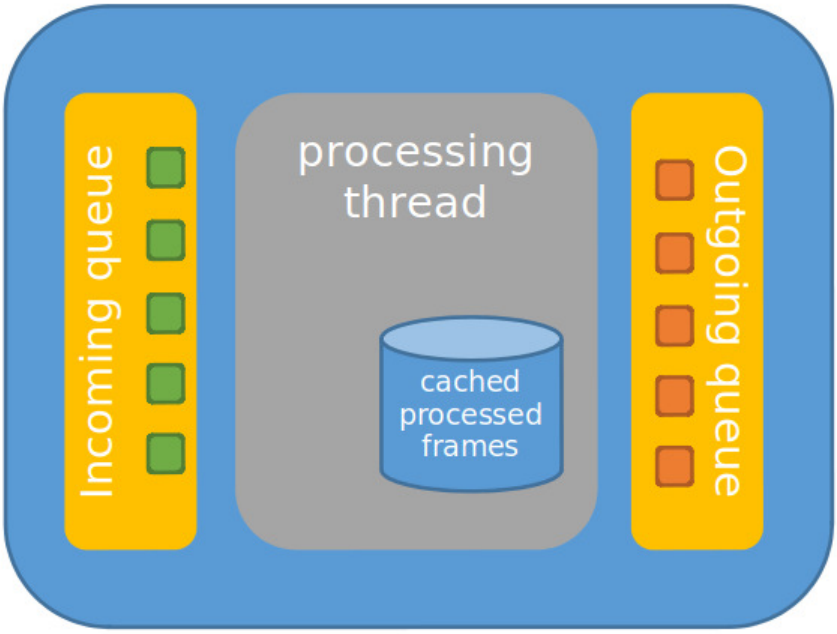

Figure 3·4: Implementation of a VF.

3.2.1 Model Formulation

Each service is described by a VFC, and in general, a single VF can have multiple instances (replicas) within the edge environment. The rationale behind the replication

of the VFs is that when a VF requires a processing time larger than the one prescribed by the required QoS. Note that:

1. the processing time of the whole VFC is actually equal with the processing time of the most demanding VF in the VFC, and

2. the processing time of the VFs must be almost identical, due to the streaming nature of the described services.

If VF_i produces data faster than VF_{i+1} can consume, then data will flood VF_{i+1} leading to the overflow of the input queue of the relevant edge device.

When a user subscribes to a service (e.g., object detection, etc.), VFO instantiates the VFC by implementing the following tasks:

1. Calculates the required number of instances (replicas) for each VF, in order to meet the service's QoS criteria,

2. Allocates the VFs to edge devices and

3. Establishes communication channels between the edge devices.

A VF instance can be part of several service chains (Fig. 3·5). Table 3.1 summarizes the formulations of the main entities of the proposed modeling framework. More specifically, $\overrightarrow{S}$ refers to a decomposed service, aiming to be deployed, $\overrightarrow{VF}$ describes the features of a single virtual function, $\overrightarrow{K}$ refers to a processing edge node and finally $\overrightarrow{W_{ab}}$ refers to the communication channel between nodes a and b.

It is important to mention that we consider as the primary component of the QoS the number of frames the service can successfully process per second. This processing frame-rate influences the data volume injected in the edge network and thus is is correlated with the capacity of the distributed network to undertake specific load. As there are additional components that could be considered, as the resolution of

the frame, we chose not to consider them for defining the QoS, due to the fact that frame-rate has a much higher contribution to the data volume.

Finally, it should be stressed that while for the video surveillance scenario we will consider processed frames/sec for quantification of QoS, but other metrics can be used as well, depending on a specific usage scenario. For example, if a service aimed at collecting and assessing the sentiment from social media feeds, the QoS could be defined as the number of messages processed/sec.

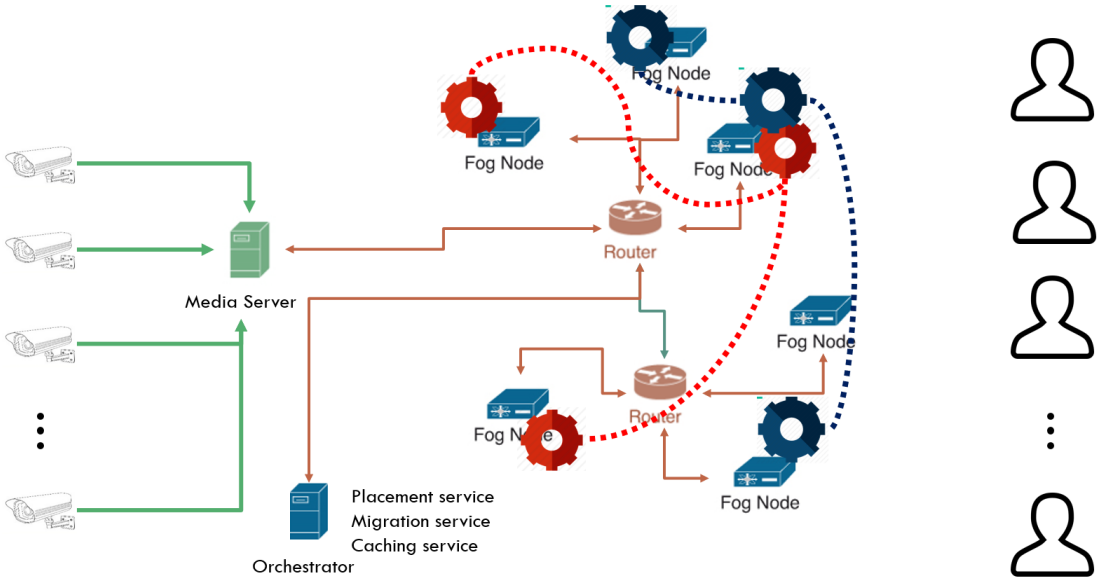

Figure 3·5: Conceptual architecture of the proposed VFC model.

3.2.2 Problems definitions and solutions

VFO node needs to assign the VFC to the edge environment. In order to achieve this, the following general assignment constrained problem needs to be solved:

Problem 1: *Determine the number of VF instances (replicas) and assign each instance to an edge device, such that the video analytics are generated while maintaining the required fps and the total network cost be minimum.*

Problem 1 can be formulated as:

Table 3.1: Basic entities of the proposed VFC model.

Entity	Formulation	Description
Surveillance service	$\overrightarrow{S} = [fps, \{VF_j\}]$	fps is the processed frames / sec the service requires (QoS)
		$\{VF_j\}$ is the set of processes comprise the service
Virtual Function	$\overrightarrow{VF} = [cpu_{load}, outdata]$	cpu_{load} is the required CPU instructions per frame
		$outdata$ is amount of data virtual function produces after processing the input data
Edge Node	$\overrightarrow{K} = [m, c(l), r(l)]$	m is the CPU instructions / sec the device can execute
		$c(l)$ is the cost function of the device, when performing l CPU instructions. Cost is a general term which includes battery life, maintenance cost, etc.
		$r(l)$ is the required time to process l CPU instructions
Link	$\overrightarrow{W_{ab}} = [bw]$	bw is the communication bandwidth among edge nodes a and b

$$min\left\{\sum_{i=1}^{n}\sum_{j=1}^{y}x_{i,j}c_{i,j}\right\} \tag{3.1}$$

$$\sum_{j=1}^{y}x_{i,j}=1 \tag{3.2}$$

$$\sum_{i=1}^{n}x_{i,j}\leq 1 \tag{3.3}$$

$$time_{computational}+time_{communication}\leq\frac{1}{fps} \tag{3.4}$$

, with

$$min\left\{y=\sum_{j=1}^{m}instances_j\right\} \tag{3.5}$$

$$time_{computational}=\max_{i=1..n}^{j=1..y}\{x_{i,j}t_{i,j}\}$$

$$time_{communication}=\max_{i,i'=1..n}^{j=1..y-1}\left\{x_{i,j}x_{i',j+1}\frac{output_j}{W_{i,i'}}\right\}$$

, where n is the number of edge devices, m is the number of the VFs, y is the total number of VFs, $x_{i,j}=\begin{cases}1 & \text{if } VF_j \text{ is assigned on node } n_i\\ 0 & \text{otherwise}\end{cases}$, $t_{i,j}$ is the required time for device D_i to execute VF_j and process the data produced from a single frame and $instances_j$ is the number of the required instances for VF_j. Finally, $W_{i,i'}$ is the bandwidth of the communication link between $node_i$ and $node_{i'}$, which host two adjacent VFs, j and $j+1$. This formulation describes a model which aims to minimize the total cost of the service (eq. 3.1) while meeting the QoS constraints (eq. 3.4), with the minimum number of VF instances (eq. 3.5), requiring that each VF instance must by assigned to exactly one edge device (eq. 3.2) and each edge device can undertake no more than one VF instance (eq. 3.3).

This is a NP-Hard problem (Besse and Chaib-draa, 2007), which requires a sub-

stantial computational time to be solved. Problem 1 is actually an variation of the Job-Shop problem (Manne, 1960), which can be reduced to the Traveling Salesman Problem (TSP) (the cities are the machines and the salesman is the job). Since TSP is an NP-Hard problem, Problem 1 is also an NP-Hard problem.

In order to acquire a feasible solution within an acceptable time-frame, we decouple *Problem 1* to two sub-problems: (A) VF instances sub-problem and (B) the assignment (placement) sub-problem.

Sub-problem A aims to identify the minimum number of instances for each VF. As discussed in the previous section, one instance from each VF must be deployed on the VFC, in order to support the service. Let $GVF = [VF_1, VF_2, ..., VF_n]$ be the set of the first instances of each VF. Each one of these VFs will be deployed to a different edge device. At this point of the assigning workflow, the allocation cost is not considered. Yet, we seek if there is a feasible solution of the placement, such that the QoS constrain is met on a specific edge instance. This results to the following relaxation.

$$min \left\{ \max_{i=1..n}^{j=1..y} \{x_{i,j} t_{i,j}\} + \max_{i,i'=1..n}^{j=1..y-1} \left\{ x_{i,j} x_{i',j+1} \frac{output_j}{W_{i,i'}} \right\} \right\} \tag{3.6}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \tag{3.7}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \tag{3.8}$$

Regarding $t_{i,j}$, it can be calculated using the $r_d()$ function of the edge device d. Thus $t_{i,j} = r_i(N_j)$, where N_j refers to the CPU instructions required by VF_j to complete its task.

Equation 3.7 reflects the fact that each VF from the GVF set must be appointed

only to one node and (eq. 3.8) that each edge node can not undertake more than one VF. *Sub-problem A* can be solved in a polynomial time by modeling it as a Min Cost Flow problem (Ahuja et al., 1988). The utilized solver is a typical Hungarian algorithm. This process results to an allocation of the GVF with the minimum required time that the network can support. Let t^* be the resulting time. If $t^* \leq \frac{1}{fps_{serv}}$, then the edge network can support the service without having to replicate a subset of the VFs. In this case, we can re-formulate the assignment problem as a constrained mixed integer problem, with the following formulation:

Sub-problem B:

$$min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} c_{i,j} \right\} \tag{3.9}$$

$$\sum_{j=1}^{m} x_{i,j} = 1 \tag{3.10}$$

$$\sum_{i=1}^{n} x_{i,j} \leq 1 \tag{3.11}$$

$$\max_{i=1\ldots n}^{j=1\ldots y} \{x_{i,j} t_{i,j}\} + \max_{i,i'=1\ldots n}^{j=1\ldots y-1} \left\{ x_{i,j} x_{i',j+1} \frac{output_j}{W_{i,i'}} \right\} \leq \frac{1}{fps_{serv}} \tag{3.12}$$

The objective function (eq. 3.9) of this formulation aims to minimize the total cost of the VFC deployment to edge network, while fulfilling the QoS constraints of the service (eq. 3.12) and assigning all VFs to a device (eq. 3.10) while limiting the number of deployed VFs to a device (eq. 11). This problem, can be solved by utilizing Constrained Programming (Laborie et al., 2018) in a polynomial time.

Solving Problem 1

If $t^* > \frac{1}{fps_{serv}}$, then the computational capacity of the edge devices is insufficient to support the service's QoS, if only one instance of each VF is deployed. In order to

tackle this, we draw inspiration from the recently launched concept of Cloud-native functions (Aderaldo et al., 2019), which handle dynamically the number of their instances aiming to handle the incoming requests. Thus, a mechanism is proposed, according to which a VF can be launched to multiple devices, and share the data coming from the previous VF of the VFC following a routing algorithm. For the work presented in the following sections, unless stated otherwise, for routing algorithm, the route-robin has been deployed, especially due to its simplicity and fairness.

According to this mechanism, VF replicas are deployed to different nodes, and each replica undertakes a portion of the data produced by the previous VF on the chain.

Thus, if a second instance of VF_j is deployed, the required time for the function to process the data related to a single frame changes from $r_k(l)$ to $\frac{r_k(l)}{2} + b$, where b is the time overhead implied for handling the data separation on VF_{k-1} (previous VF in the chain) and data merging on VF_{k+1} (next VF in the chain), assuming that the two instances of the VF is deployed to identical nodes.

This case leads as to the formulation of a new sub-problem (*sub-problem C*). Its objective is to identify the minimum number of replicate instances for each VF that need to be deployed on the edge network, in order to meet the QoS constraints. Let $\overrightarrow{S} = [s_1, s_2, ..., s_m]$ represent the number of instances for each one of the VFs, with s_i being an integer larger or equal to one ($s_i \geq 1$). Thus, we derive the following problem formulation:

$$min \left\{ \sum_{j=1}^{m} s_j \right\} \tag{3.13}$$

$$\sum_{j=1}^{m} \left(\frac{r_f(N_j)}{s_j} + (s_j - 1)b + \frac{output_j}{W_{ff'}} \right) \leq \frac{1}{fps_{serv}} \tag{3.14}$$

, where N_j are the required instructions per frame required for executing VF_j

on device f, with f' undertaking VF_{j+1}. Equation 13 drives our model to produce solutions with the minimum total new instances of the VFs, while (eq. 14) satisfies the QoS of the surveillance service. Unlike the problem formulated by *eq. (3.9-3.12)*, this is a non-linear mixed integer problem, which requires the utilization of the active set solver APOPT (Hedengren et al., 2012). Let the result of this sub-problem be $instances = [ins_1, ins_2, ..., ins_m]$. Using $instances$, we can revisit *Sub-problem B* and solve the allocation problem as before.

The discrepancy of the latest described sub-problem is that the utilized nodes f and f' are unknown. This is rational, because we seek the number of the VF instances with regard to the computational time, which depends not only on the VF load but also on the node that will undertake the VF. We resolve this deviation by using Algorithm 1. This algorithm can provide two approaches:

(i) worstCase scenario, where the edge device used to calculate (eq. 3.14) is the one with the lowest processing capacity and

(ii) bestCase scenario, where the highest processing capacity device is utilized.

Algorithm 1 receives as input the processing fps implied by the QoS and the allocation of the initial instances of the VFs. As depicted in Fig. 3·6, both approaches converge to the desired processing fps. The reported results have been derived by a simulation framework that has been developed in order to evaluated the reported approach (simulation plots) and by an experimental edge network setup (system plots). Details on the utilized edge environment are provided in Section 3.3.2.

As far as the two functions ($addReplicate()$ and $removeReplicate()$) used in Algorithm 1, they calculate for each VF either the improvement (for the $addReplicate()$) or the regression (for the $removeReplicate()$) a new instance will have to the total cost. Let $\{v_i\}$ be these values. Then, we choose the VF which minimizes the difference $|fps_{current} - v_i|$. In each iteration, *Sub-problem B* is solved.

Algorithm 1: Online placement optimization algorithm

```
Input: fps, x_ij, ε (error tolerance)
deploy(x_ij);
fps_current = measure_fps();
if |fps_serv − fps_current| < ε then
    return x_ij
else
    while (|fps − fps_current| > ε) do
        sleep(1);
        if (fps_real < fps_current − ε) then
            x_ij = addReplicate(x_ij)
        else if (fps_real > fps_current + ε) then
            x_ij = removeReplicate(x_ij)
        else
            return x_ij
        end
        deploy(x_ij);
        fps_current = measure_fps();
    end
end
```

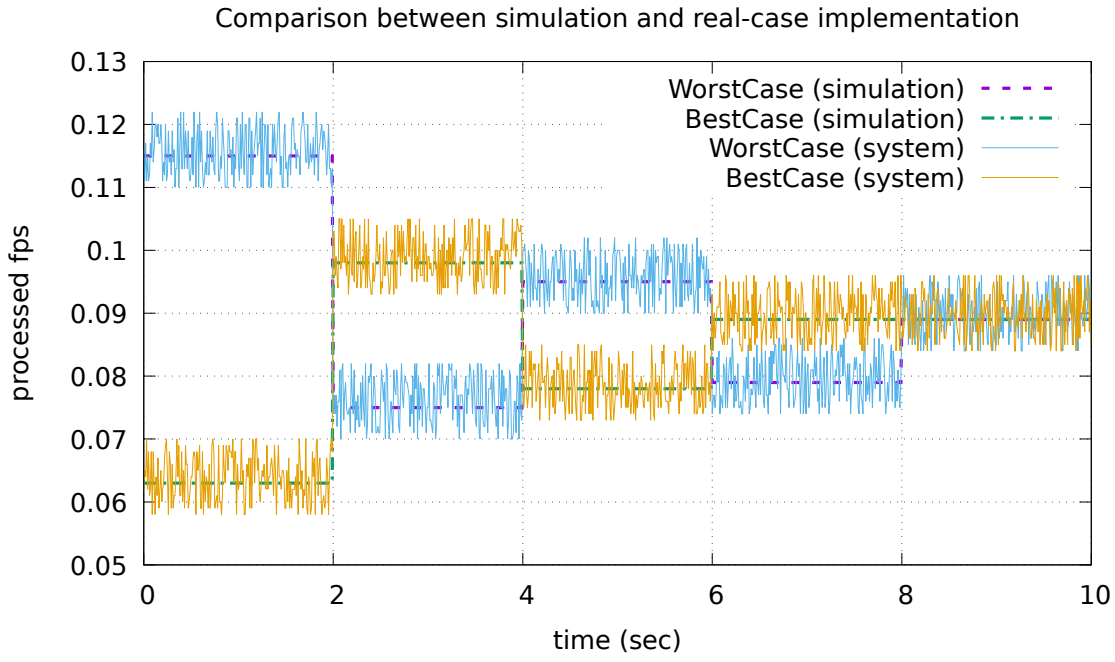

Figure 3·6: Algorithm 1 comparison between modeling and real-case system implementation.

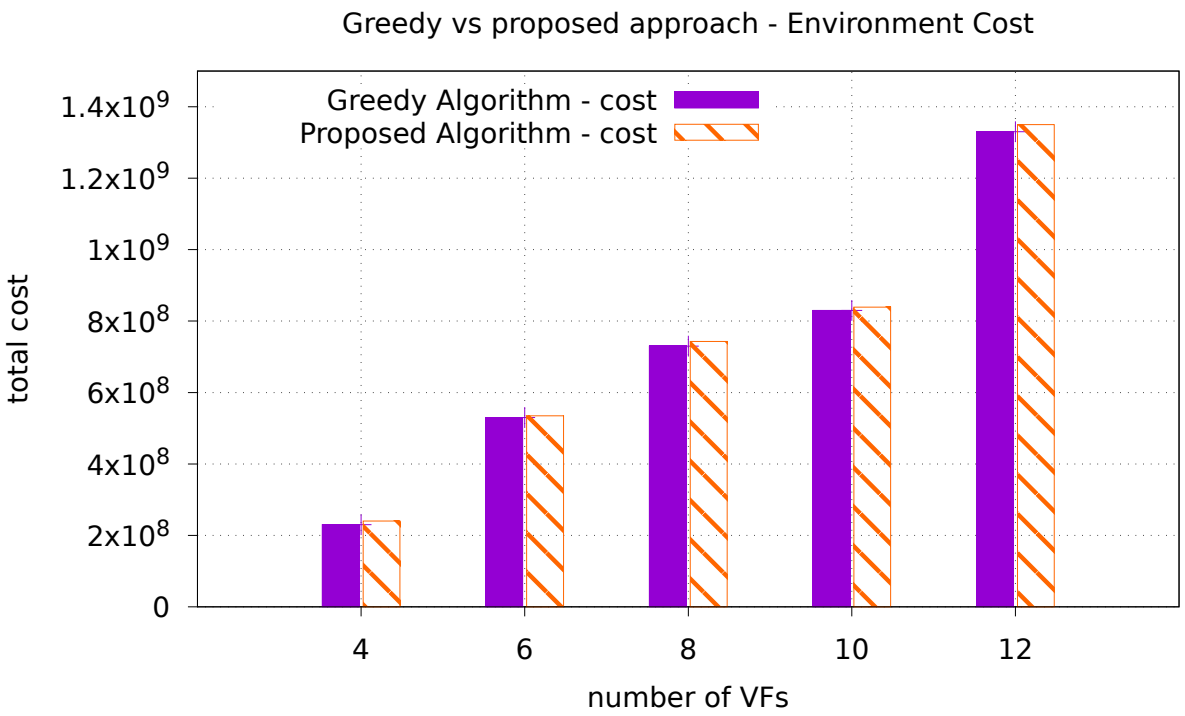

Figure 3·7: Comparison between the optimum solution (greedy algorithm) and the proposed algorithm - Environment cost.

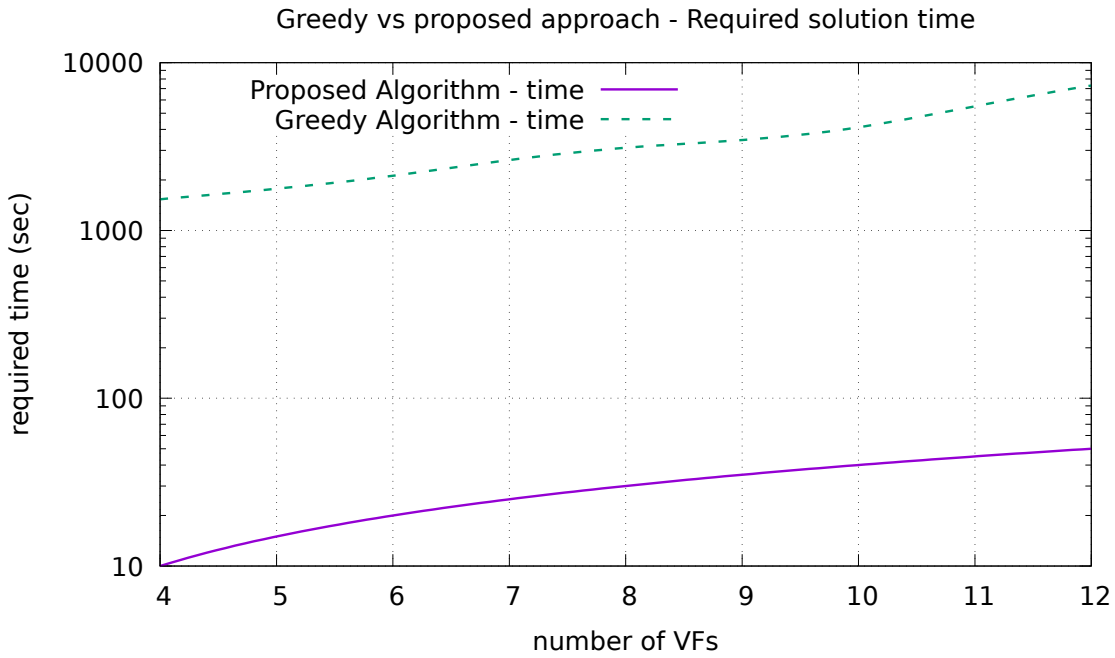

Figure 3·8: Comparison between the optimum solution (greedy algorithm) and the proposed algorithm - Required solution time.

Aiming to evaluate the accuracy of the proposed algorithm for solving *Problem 1*, a set of scenarios (edge networks and $VFCs$) have been setup, solving *Problem 1* both using a naive greedy algorithm (which calculates the optimum solution) and the proposed approach. More specifically, the greedy algorithm exhaustively creates placement solutions and calculates the total cost for each solution. After testing all possible combinations, the algorithm selects the solution with the minimum total cost. For the number of replicas, the greedy algorithm tests all scenarios with the number of replicas taking values from 0 to $n - m$, where n is the number of edge nodes and m the number of the different VFs.

Five different $VFCs$ have been used. For each VFC, 15 scenarios have been established by setting the parameters presented in Table 3.2. The parameters were acquired from the experimental setup, after following the measurement approaches suggested in (HajiRassouliha et al., 2018) and in (Zou et al., 2009). Figures 3·7 and 3·8

present the results of this comparison. The reported results are the average values of the total cost among the 15 different scenarios. The total average extra cost imposed by the proposed approach was $\approx 1.97\%$, while the required time for each algorithm to solve the problem is $\approx 5.61sec$ for the proposed approach and $7.89 \times 10^3 sec$ for the greedy algorithm (Fig. 3·8). The solvers, which have been implemented using GEKKO suite (Beal et al., 2018), have been executed on a Intel i7 2.8GHz (8core) on 8GB of RAM.

Table 3.2: Parameters on experiment setup.

Parameter	**Value**
cpu_{load}	$10^4 N(10, 0.8)$
m_k	$10^3 N(20, 0.9)$
Wtt'	$10^5 N(10, 0.7)$
$output$	$10^3 N(200, 0.65)$

3.2.3 QoS monitoring and failure avoidance

The model described in the previous sections is used in order to initialize and instantiate a VFC for serving a surveillance service. Yet, during the execution time of a service, the edge environment, unlike cloud infrastructures, is highly dynamic. The edge devices, due to low resources, appear fluctuations in their main performance metrics, like available CPU and RAM.

A surveillance service, in order to maintain the QoS standards, adequate resources are required through its lifecycle. The proposed VFC model is more prune to the fluctuations on the performance indicators compared to the $Monolithic$ approach, as it depends not only from one edge device but from a set of edge devices, where if one of the hosting devices fail (overload, battery drain, network disconnection), the whole service collapses.

Aiming to address this issue, a *"failure alert"* methodology has been designed and developed, based on a well established Recursive Neural Network, the Long-Short

Term Memory (LSTM). By *failure* we consider the overloading of an edge node at such level that the assigned VF can no longer be executed properly, in terms of assigned mips (million instructions per second).

A variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by (Cho et al., 2014). It combines the forget and input gates into a single "update gate" while it also merges the cell state and hidden state compared to the classic LSTM cell. The core memory cell of the utilized network is presented in Fig. 3·9 and governed by (eq. 3.12) - (eq. 3.18).

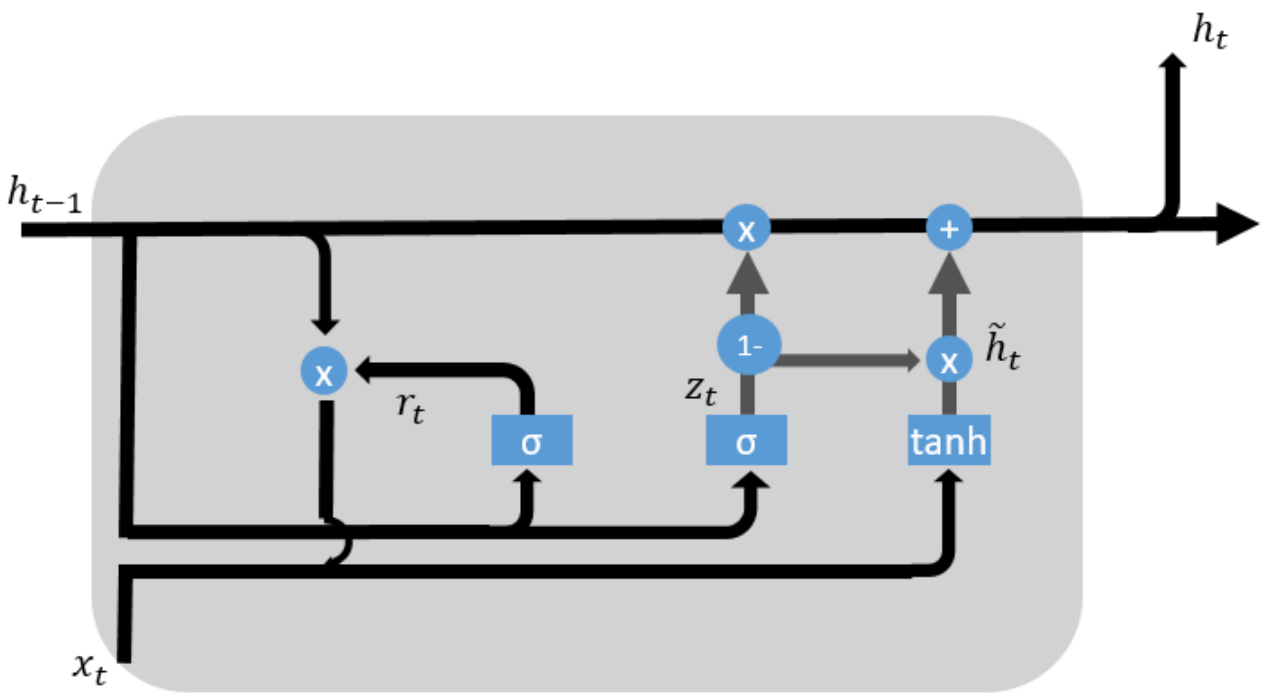

Figure 3·9: LSTM memory cell.

$$z_t = \sigma(W_z) \cdot [h_{t-1}, xt] \tag{3.15}$$

$$r_t = \sigma(W_r) \cdot [h_{t-1}, xt] \tag{3.16}$$

$$\widetilde{h_t} = \tanh(W \cdot [r_t * ht - 1, x_t]) \tag{3.17}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t} \tag{3.18}$$

The model learns long term dependencies on the performance metrics of the edge devices. More specifically, two LSTM models have been established, one for predicting CPU usage and one for predicting RAM usage. The training datasets have been created using the benchmark edge environment and by mimicking artificial fluctuations on the edge devices. Details on the process are provided in Section 3.3.3. When the inference of the model (which is executed in the VFO) predicts high CPU and / or RAM utilization, it informs the relevant module for the specific VFC that is possible to face a failure within the specific time-frame. At this point, VFO recalculates the optimal placement of the VFC and resets the VFC.

3.2.4 Caching Mechanism

Surveillance as a service is one of the most promising models for delivering surveillance analytics to the end users. According to this model, a user can choose a camera feed and request for specific analytics. It is important to consider that the VFC model enables the sharing of the results among different services, in case two (or more) services share the same(s) VFs. For instance, a municipality offers video analytics services on live video streams from the busiest shopping streets of a region. A user can request a service named "Count women" from a specific video camera for two weeks, as she / he is interested on opening a beauty salon while another user request a service named "Detect abandoned items" from the same video stream. While the two services seems to have nothing in common, they share a subset of common VFs, *image-enhancement* and *light-equalization*.

For this, a caching mechanism has been designed and developed, according to which when a node executes a VF for data related with frame k, it stores the results in a stack for a specific time-frame. In case another video analytics service request from the same VF to process data related to an already processed frame, it retrieves the results and forwards them to the next VF of the VFC, without recalculations.

3.3 System implementation

Aiming to evaluate the VFC model described in Section 3.2, both simulation and prototype platforms have been developed, where all the necessary functionalities have been deployed to support AI real-time video analytics of surveillance services. More specifically, three discrete tools have been formulated: (i) a simulation platform, (ii) an edge benchmark environment and (iii) a cloud-based surveillance service.

3.3.1 Simulation Platform

The simulation platform has been developed under Python 3.6 (Gries et al., 2017). All the entities described in Table 3.1 have been modeled as distinct processes. Linux commands *cpulimit* and *ulimit* have been utilized to mock specific computational capacities for each 'edge device'. In the experiments described in the Section 3.4, each video analytic service has been modeled as a set of n VFs, where n is a random integer $\in [3, 7]$. Each VF could be either a *light* VF, a *moderate* VF or a *heavy* VF, with relative computational characteristics. Finally, each VF may by identical with another VF with a probability of 15%, enabling the caching mechanism described in the following sections.

Withing the simulation platform, three different *Setups* of a surveillance service have been implemented.

- **Setup I**: The surveillance service has been implemented under a monolithic

approach. This means that each deployed surveillance service has been hosted in one edge device (details in 3.3.4).

- **Setup II**: The surveillance service has been implemented under a simplistic distributed method, where each surveillance service were deployed to multiple edge devices (details in 3.3.4).

- **Setup III**: The proposed VFC model.

All of the above setups have been tested under different service demand probability distributions. By service demand probability distribution, we refer to the probability a user requests a service at a specific time-point t_d. More specifically, various Poisson distributions have been used. For the Poisson distributions, three different λ parameters have been used, aiming to mimic low, normal and high user demand rates, according to (Hossain, 2011), (Song et al., 2014) and (Nwokolo and Inyiama, 2017). For the same Setup, the cumulative number of the requested services was the same. All the different scenarios are presented in Figure 3·10.

3.3.2 Edge environment implementation

The implemented edge network comprise 6 Raspberry PI 3 (model B+) devices, with a Quad Core 64bit CPU @ 1.2GHz and 1GB RAM and 2 Raspberry PI 4 devices with a Quad core Cortex-A72 (ARM v8) 64-bit CPU @ 1.5GHz with 4GB RAM, running Raspbian OS. The feed from the camera has been mocked as video file from the VIRAT dataset (Oh et al., 2011).

Two video analytics services have been deployed on the edge environment. *Service A* and *Service B*, require gender and age classification respectively. For the main inference model, the pre-trained deep learning models proposed in (Blog, 2018) have been used. Both *Service A* and *Service B* decompose to 4 VFs. *Service A* includes

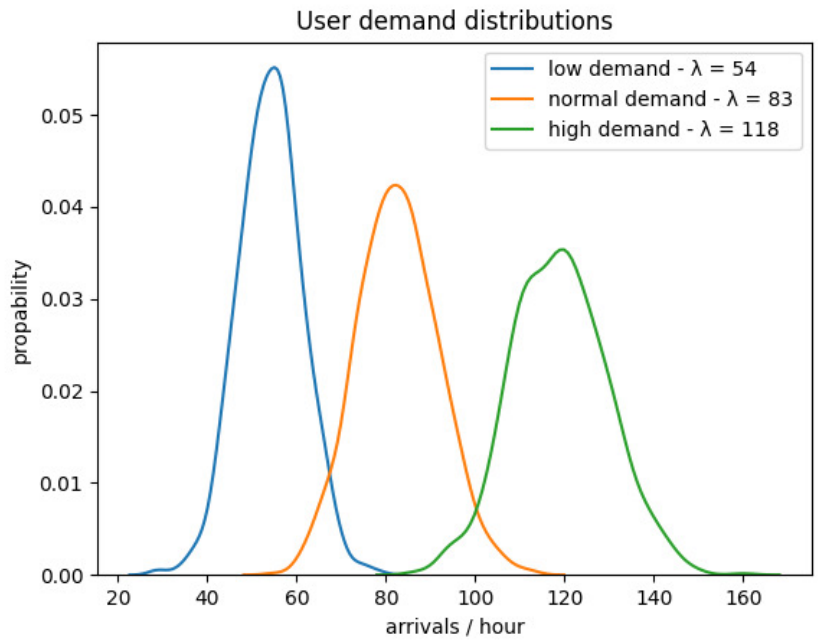

Figure 3·10: Probability distributions for user demand.

$VF_1()$, $VF_2()$, $VF_3()$ and $VF_4()$, while *Service B* includes $VF_1()$, $VF_2()$, $VF_5()$ and $VF_6()$.

Details on the VFs are presented in Table 3.3.

- $VF_1()$: Frame acquisition and image enhancement (histogram equalization and Multi-scale retinex on low light conditions,
- $VF_2()$: Blob calculation for a specific frame,
- $VF_3()$: Convolutional Neural Network (MobileNet v2) pass and probability matrix acquisition,
- $VF_4()$: Coordinates calculation for the detected objects and non maxima suppression for overlapping objects,
- $VF_5()$: Convolutional Neural Network (gender CNN networks) pass and probability matrix acquisition,

Table 3.3: Model's parameters.

Parameter	Value
n	8
W	$98 \pm 1.7Mbps$
cpu_{load}	$[10^3, 5 \times 10^3, 10^6, 10^2]$ instructions/frame
$output$	$[10^5, 3 \times 10^4, 10^7, 10^4]$ bytes
$c_k(l)$	$\frac{l^2+0.8}{1000}$
$r(l)$	$\frac{20l+9}{m_k}$ sec
m_1	10^4 instructions/sec (PI 3)
m_2	10^7 instructions/sec (PI 4)

- $VF_6()$: Results reporting.

The model's basic parameters are presented in Table 3.3.

The values have been selected after performing a set of experiments for different workloads. A non-linear model for the cost function has been chosen. VFs have been implemented using Python3, utilizing the multiprocessing library.

3.3.3 LSTM models

As discussed in Section 3.2.3, a QoS monitoring and failure avoidance mechanism has been developed, in order to enable the hosting of demanding surveillance services throughout their life-cycle. The basic concept is based on the idea of monitoring the basic performance indicators of an edge device and trying to predict the value of these indicators within a specific time-window in the future. For this prediction, two LSTM models have been utilized, one for each performance indicator. While the possibility of using a single LSTM model has been explored, by combining RAM and CPU utilization in a singe metric, the experimental results have shown that it is more appropriate to deploy two models, one for each metric.

As far as the architecture of the deployed LSTM models is concerned, a two layer approach has been adopted, with one hidden dense layer, each of them comprise 100

nodes. ReLU has been used as the activation function and ADAM solver for the optimization steps. Finally, the Mean Square Error has been selected as the loss function.

The challenging part of this approach is to acquire a suitable dataset for successfully training the LSTM models. As no suitable dataset came to our knowledge, the edge environment described in 3.3.2 has been used in order to produce the suitable datasets.

An agent hosted in the VFO device constantly collecting data regarding the CPU and RAM utilization for each device which is part of a VFC. A software which can mimic overload demand on the edge devices (stress tool) has been installed on each edge device, under a random distribution on the required resources. At the same time, VFO monitors the QoS (processed frames / sec) for each one of the deployed services (Fig. 3·11).

The training dataset has been created after 248 hours of monitoring and collecting data from the eight (8) devices of the edge environment, with an interval of five (5) secs. This process has resulted to a set of time series $t_{cr} = c_i, r_i$, one for each VFC service applied on the edge environment.

The models have been implemented by taking 100 neurons in the LSTM layer. The utilized loss function is Mean Squared Error. Train and test errors are presented in Fig. 3·12 and Fig. 3·13.

3.3.4 Comparison setups

Baseline VF allocation algorithm

The first comparison study refers to the quantification of the improvement the VFC placement algorithm provides on an edge environment upon the deployment of a VFs set. To achieve this, a simple placement algorithm (Algorithm 2) needs to be considered, in order to assess the influence of the proposed placement algorithm. The

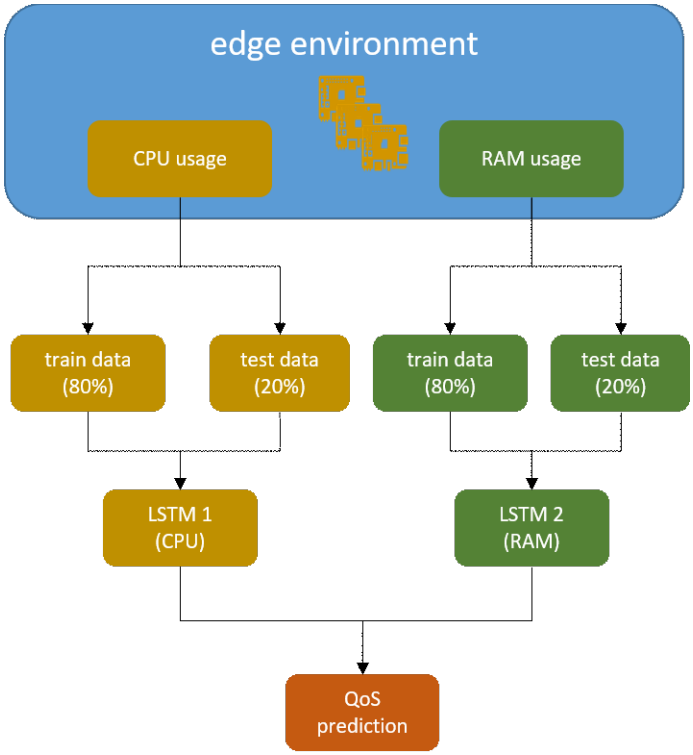

Figure 3·11: LSTM dataset creation.

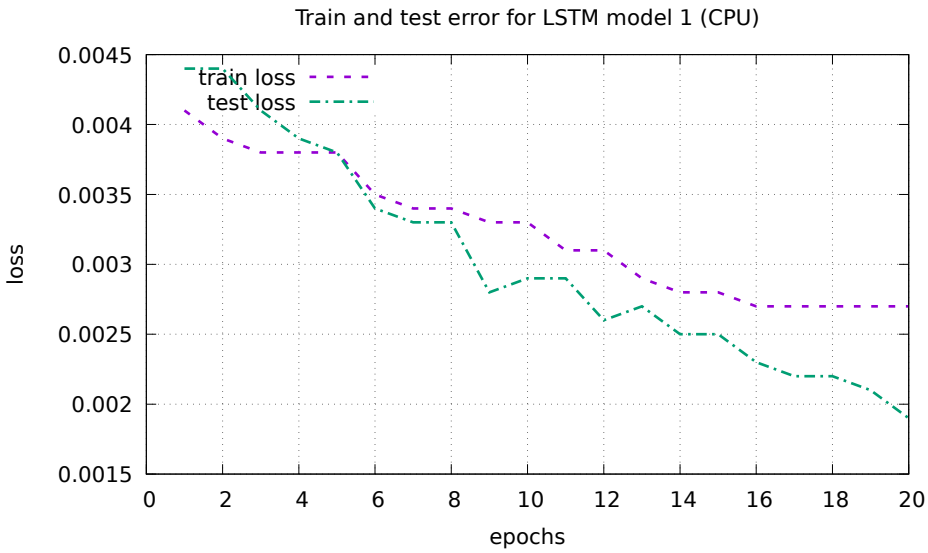

Figure 3·12: Train and test error for LSTM model 1 (CPU).

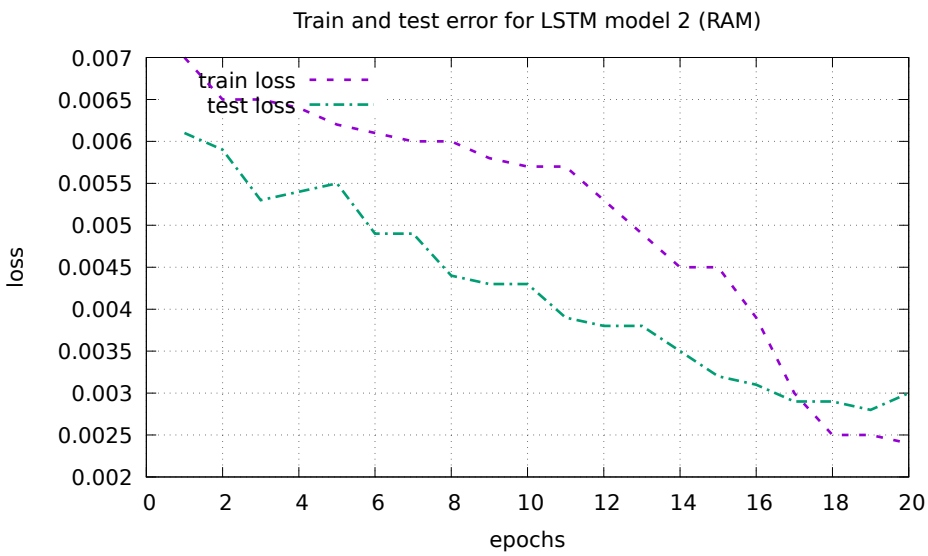

Figure 3·13: Train and test error for LSTM model 2 (RAM).

rationale of the baseline algorithm is based on an initial random placement of the VFs of a VFC on the edge nodes. After the constitution of the chain, the processed fps are measured and compared with the desired QoS. In case the QoS requirements aren't met, the baseline algorithm detects the edge node with the lowest throughput and creates a replica of the relative VF, which is again randomly placed on an edge node. The last step iterates until the QoS requirements are met, or their are no more resources to commit.

The first set of experiments concerned the evaluation of the VFC placement algorithm. For this, the simulation environment has been utilized, aiming to compare the two approaches as the edge environment scales up.

For each simulation scenario, a specific number of $VFCs$ has been considered and the simulation was executed 10 times. For each VFC, the number of VFs was randomly selected from the distribution $\lfloor \mathcal{N}(3, 8) \rfloor$. The relevant results (average values for the 10 fold executions) are presented in Fig. 3·14 and Fig. 3·15.

More specifically, one can observe that the number of the required VFs (includ-

Algorithm 2: Baseline placement algorithm.

```
Input: fps, VF_i, K_j, ϵ
x_ij = random-placement(VF_i, K_j);
fps_current = measure_fps();
if |fps − fps_current| < ϵ then
  | return x_ij
else
  | while (|fps − fps_current| > ϵ) do
  |   | sleep(1);
  |   | K_low, VF_h = getLowestThroughput(K_j) addReplicate(VF_h)
  |   |  fps_current = measure_fps();
  | end
end
```

ing replicas) are reduced at approximately 95.32% (average value) when using the VFC placement algorithm, when compared with the baseline placement algorithm. Additionally, the total edge environment cost is reduced by 68.22% with the use of the VFC placement model.

Cloud service

Aiming to compare the proposed architecture with the current common practice of deploying a surveillance service, a cloud surveillance service has been deployed. The characteristics of the utilized Virtual Machine are the following. CPU: Intel Xeon E5 v3 @2.8GHz and RAM: 32GB. The IaaS of the Google Platform ©has been chosen to host the services.

The services deployed on the Cloud are identical (*Service A* and *Service B*), in terms of implementation, with the services deployed on the edge environment.

Monolithic model

The simplest approach for deploying a video surveillance service on the edge would be to host the service on a single edge node. Despite the simplicity of this approach, several advantages can be found, like the easy deployment and the straightforward

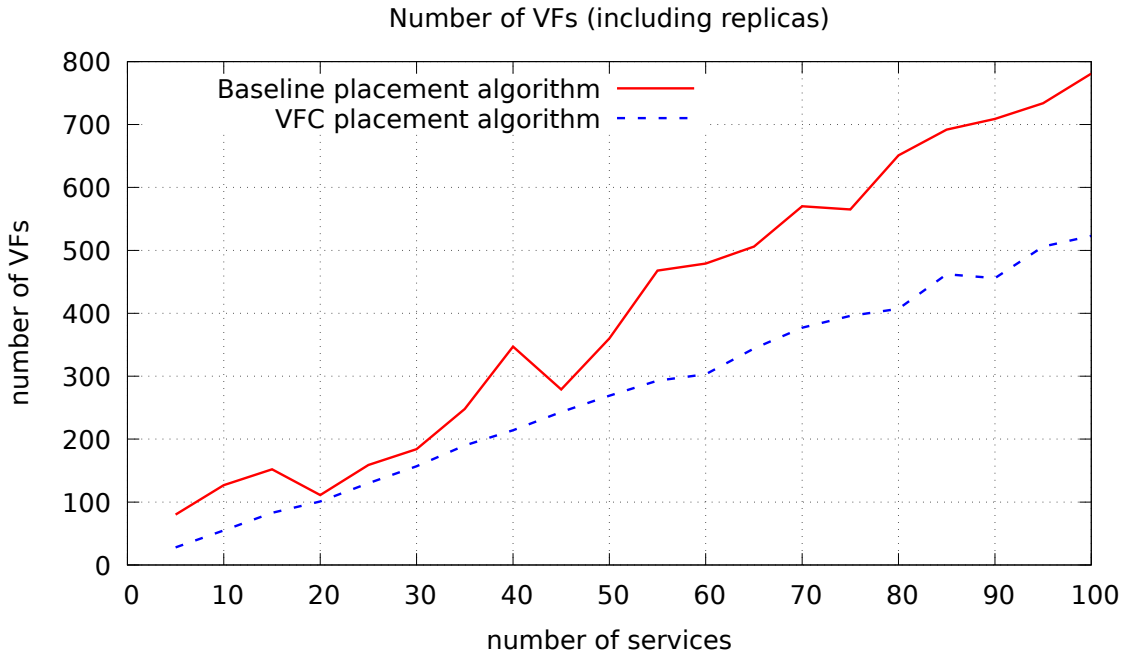

Figure 3·14: Baseline vs. VFC placement algorithms - number of VFs.

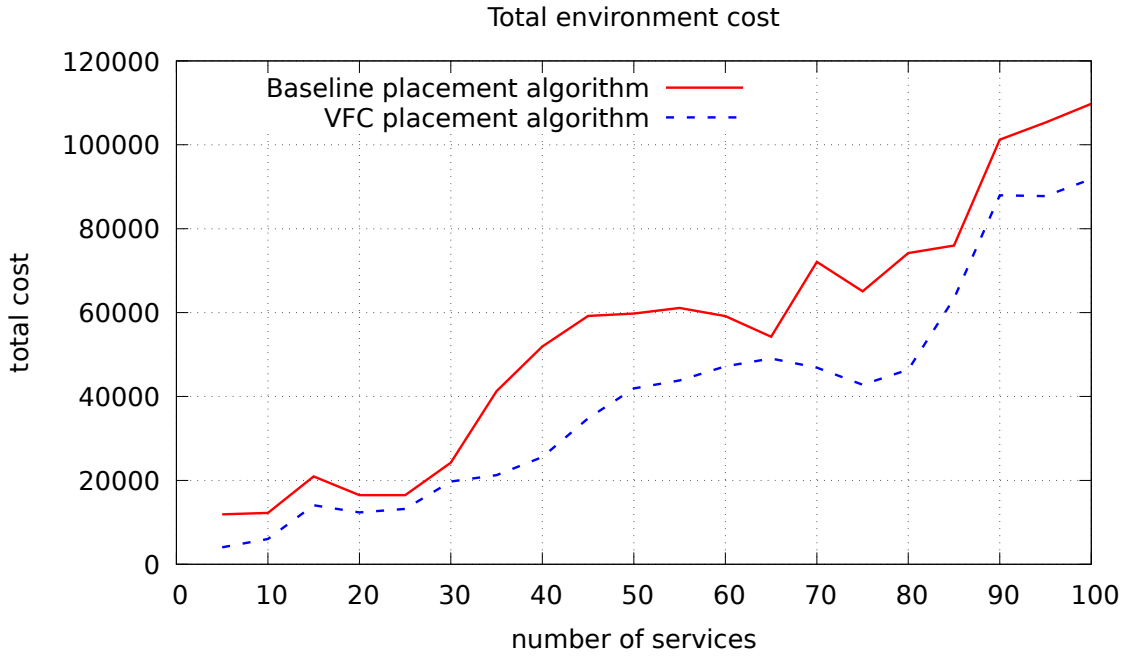

Figure 3·15: Baseline vs. VFC placement algorithms - total environment cost.

management of the service. Yet, the computational capacity of the edge nodes is expected to limit the QoS.

The specific model has been implemented and deployed to the edge environment, mainly for identifying the lowest threshold for the QoS and edge environment cost. For the implementation of the model, the same VFs have been used as the ones in the VFC model.

Simple Frame Distribution model

A second benchmark model has been implemented, also for comparison purposes. The Simple Frame Distribution model ($SFDM$) extends the $Monolithic$ model by deploying the same service on several edge nodes and by distribution the incoming frames to the nodes. This model requires an new agent which handles the packaging and distribution of the frames under a proprietary protocol and a second agent which sinks and synchronize all responses and inform the end user about the final result.

The implementation and deployment process of the $SFDM$ model is simpler than the VFC model. Yet, this approach nullifies benefits of the differential algorithms, like background subtraction (Piccardi, 2004). Nonetheless, the $SFDM$ is considered, aiming to evaluate the processing capacity of the VFC model and its cost over the edge environment.

Apache Spark framework

Apache Spark© (Zaharia et al., 2016) is a general-purpose distribution system, which can utilize the processing capacity of a cluster to perform complex computations. There are three different ways in which Apache Spark can be used for distributing computational tasks:

(i) Standalone Mode, in which Spark and HDFS (Hadoop Distributed File System) directly communicate with each other and optionally MapReduce can submit

jobs in the same cluster;

(ii) Hadoop Yarn according to which Spark executes over a Hadoop container manager distributed across the cluster and

(iii) Spark in MapReduce, where Spark can execute its own jobs along with the one submitted by MapReduce.

For benchmark purposes, the Standalone deployment mode of the Apache Spark has been selected, in which both HDFS and Apache Spark are the part of the cluster.

The master node, which acts as a server, also hosts the streaming framework of Kafka (Khochare et al., 2017), which is used to collect the input of the camera and distributed efficiently on the cluster. The Spark cluster deployed on the same edge devices as the ones used to test the VFC model. It's configuration parameters have been set according to (Nasir et al., 2019).

The acquirement of the data was performed by a single service which hosted the Apache Kafka © framework, outside the edge environment, as described in (Ichinose et al., 2017).

3.4 Experimental Setups and Results

A set of experiments have been contacted, aiming to report the performance of the proposed VFC model. One can categorize the experimental work in three parts:

- Experiments set out to assess the scalability and sustainability of the VFC model.

- Experiments set out to assess the contribution and the benefits of the two add-on services on the VFC model (caching and QoS mechanisms).

- Experiments set out to compare the VFC model with alternative frameworks which can host surveillance services.

The tools described in Section 3.3 have been used in order to evaluate the VFC model under specific metrics. Namely, the QoS of the deployed services, the total cost of the edge environment and the scaling capacity of the VFC model have been considered. A set of variants (Setups) have been used, aiming to examine the aforementioned metrics. More specifically, (as briefly mentioned in Section 3.3.1) the experimental setups described in Tables 3.4, 3.5, 3.6 and 3.7 have been implemented on the edge environment detailed in 3.3.2.

3.4.1 Assessing the scalability of the VFC model

The first metric under consideration is the scalability of the proposed model under parameters like edge devices number and user services demand. To examine these parameters, the simulation platform detailed in Section 3.3.1 is utilized.

Fig. 3·16 and Fig. 3·17 present the experimental results of the simulation environment for $Monolithic$ model, SFD model and for the VFC model. All models refer to single service implementations. More specifically, Fig. 3·17 presents the number of required edge devices in order to support a specific number of services, while Fig. 3·16 presents the total cost of the edge environment as the number of requested services increase. The characteristics of the simulated services are given in Table 3.2. It is obvious that the VFC model enables the support of a specific number of services with fewer edge devices and with substantially lower total cost, as the service demand scales.

Next, the scaling of the VFC model in a temporal simulation scenario is assessed. For this, simulations for the $Monolithic$ model and the VFC model have been implemented under a 24 hour scenario, according to which users request surveillance services under different probability distributions (Fig. 3·10). Five parameters have

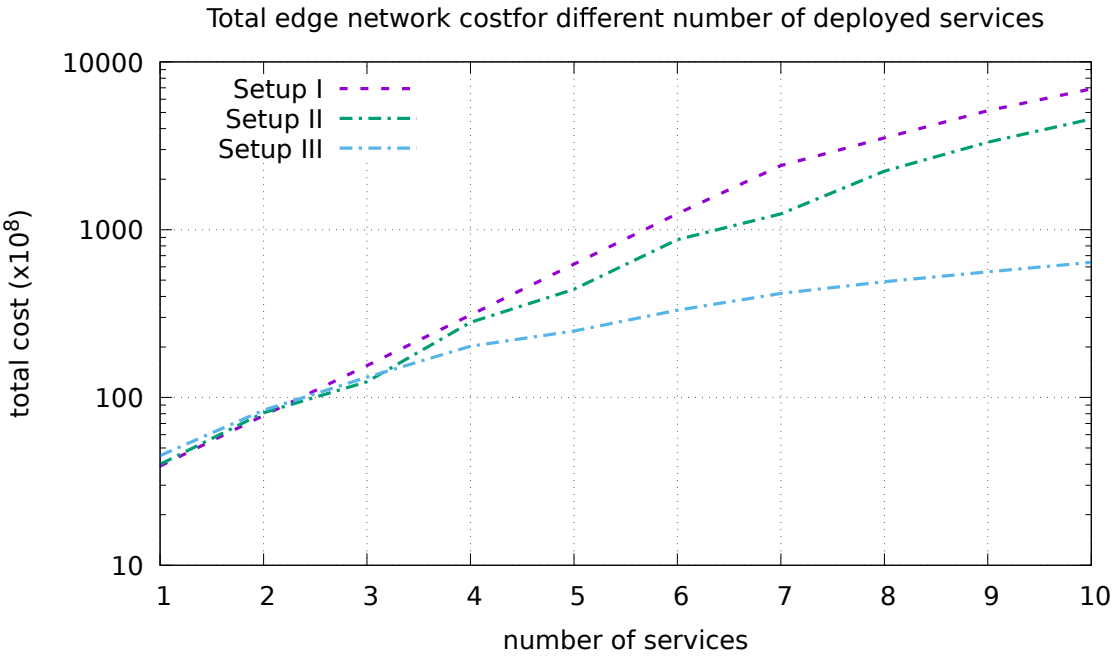

Figure 3·16: Total network cost (simulation environment).

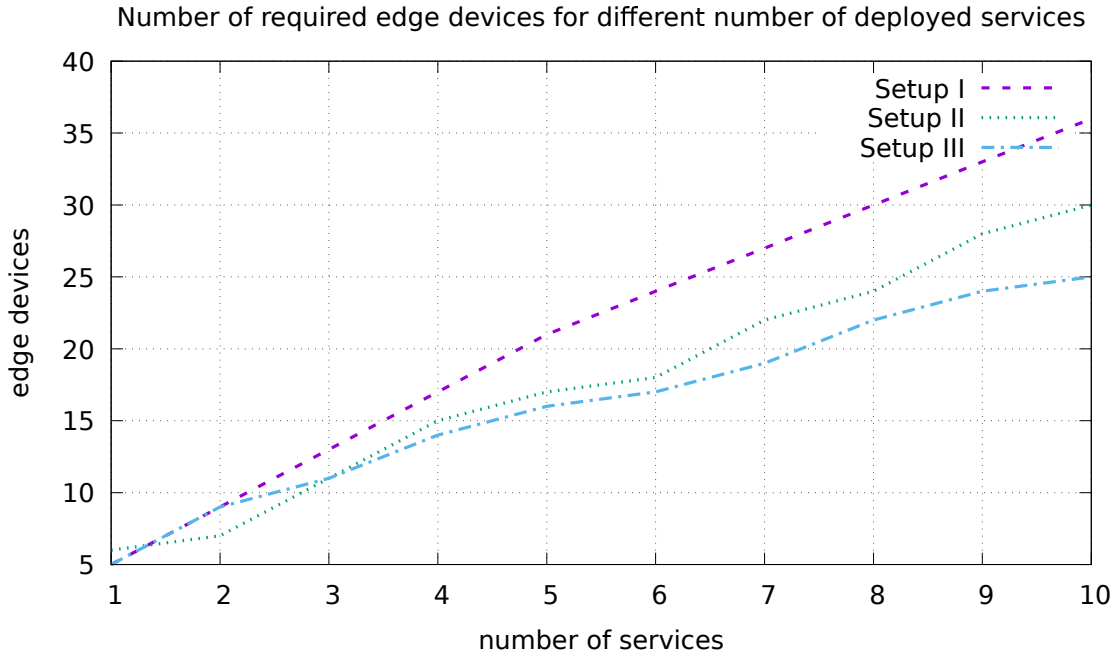

Figure 3·17: Total edge devices (simulation environment).

been examined. Namely, the percentage of served services over total requested services (Fig. 3·20), the percentage of rejected services (due to lack of resources) over total requested services (Fig. 3·19), the total edge environment utilization (Fig. 3·18), the total edge environment cost (Fig. 3·21) and the edge network traffic (Fig. 3·22) have been calculated, as the number of the edge devices scales up. It is important to mention that low demand scenarios appear to produce more traffic and higher environment cost due to the fact that fewer services are rejected during these setups.

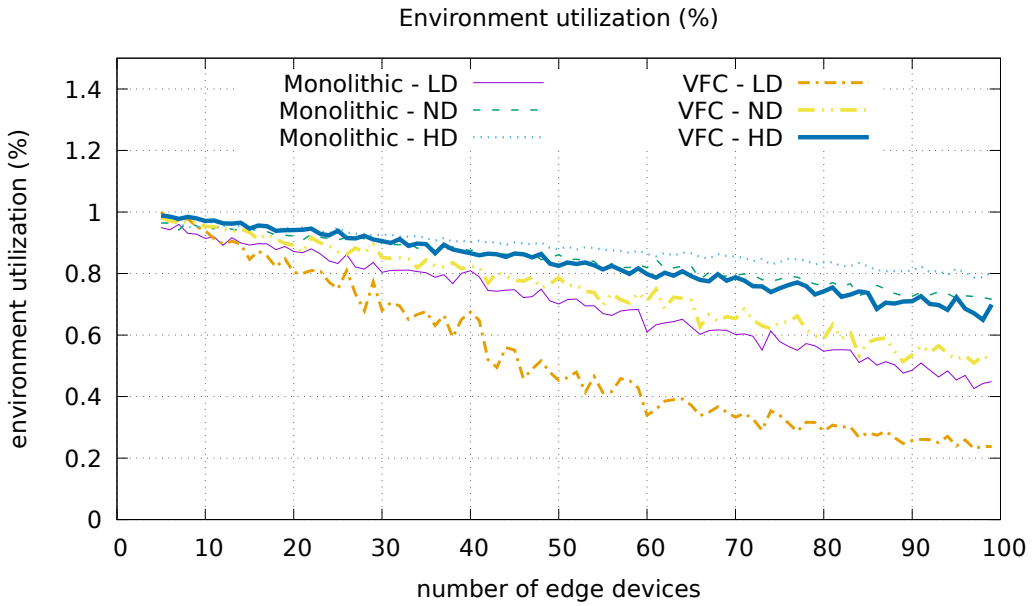

Figure 3·18: Edge environment utilization on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

Combining the results of aforementioned figures, one can notice that the VFC model can achieve higher percentages of served services, under all three of the different user demand distribution probabilities, while maintaining lower levels of total environment cost. Additionally, VFC model can achieve higher edge devices utilization while reducing the percentage of the reject services must faster as the network scales, always compared with the $Monolithic$ model.

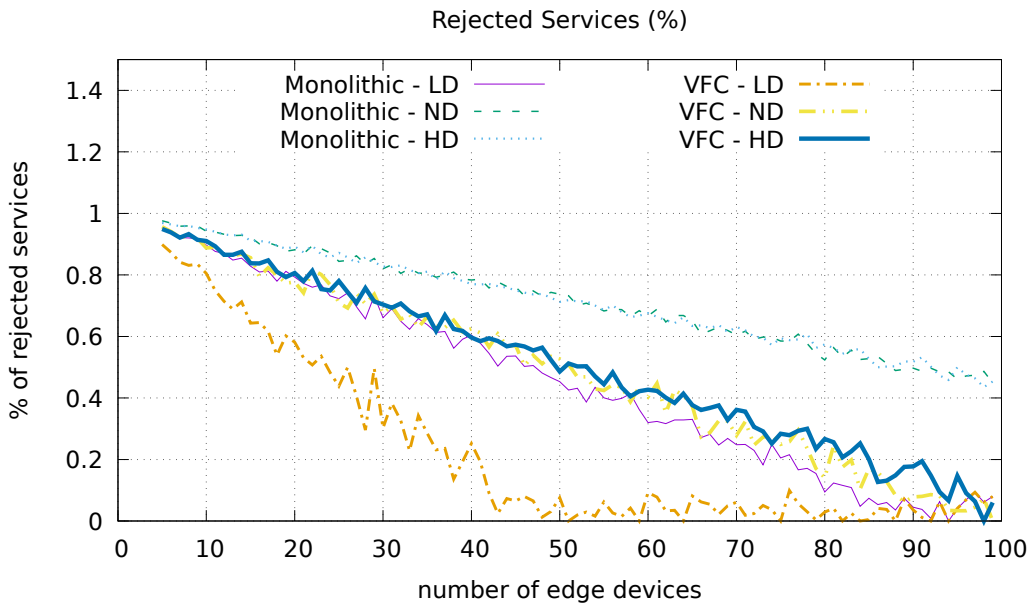

Figure 3·19: Percentage of rejected services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

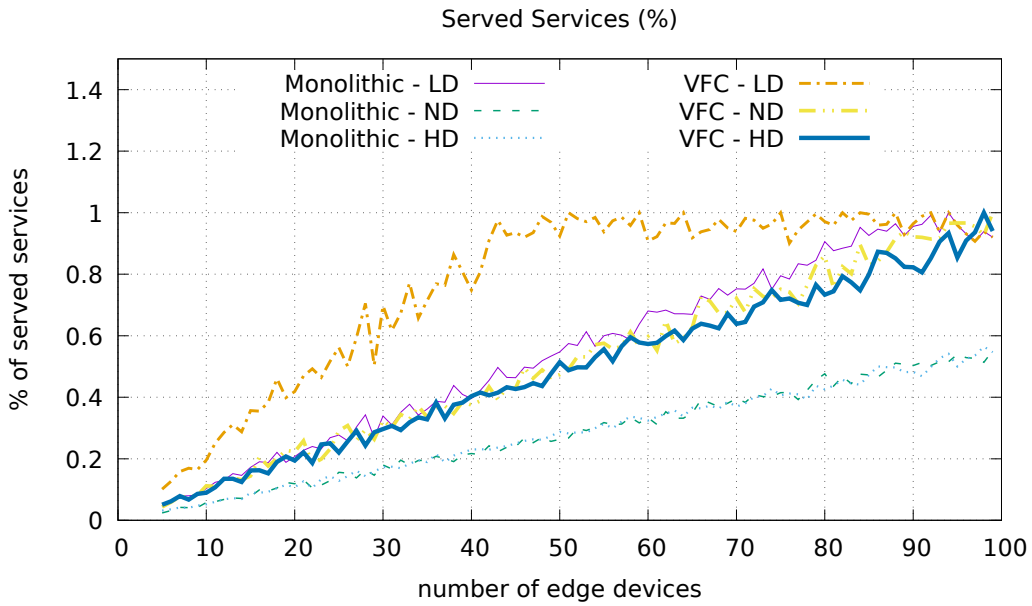

Figure 3·20: Percentage of served services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

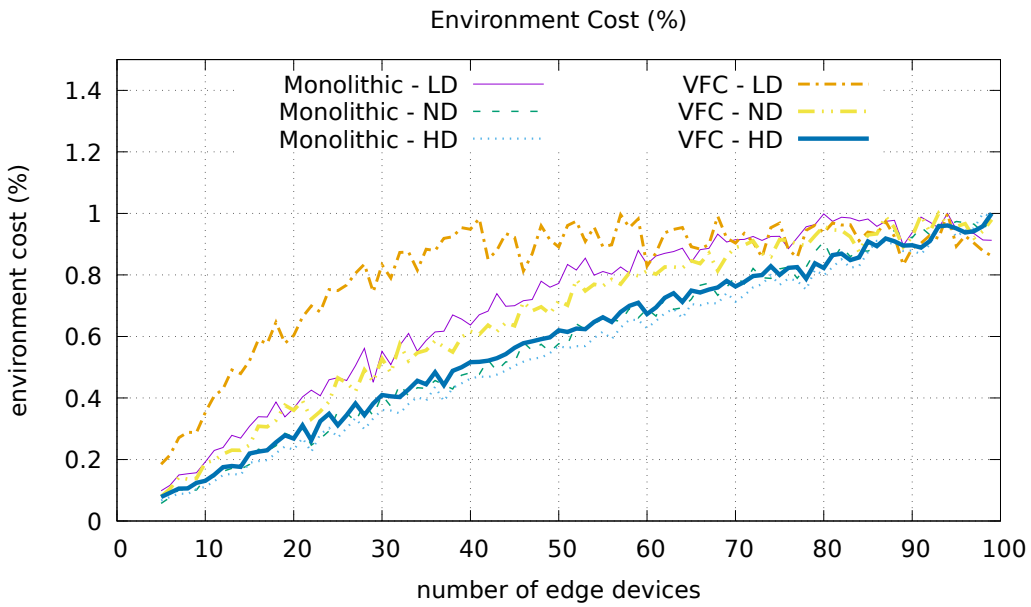

Figure 3·21: Edge environment cost on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

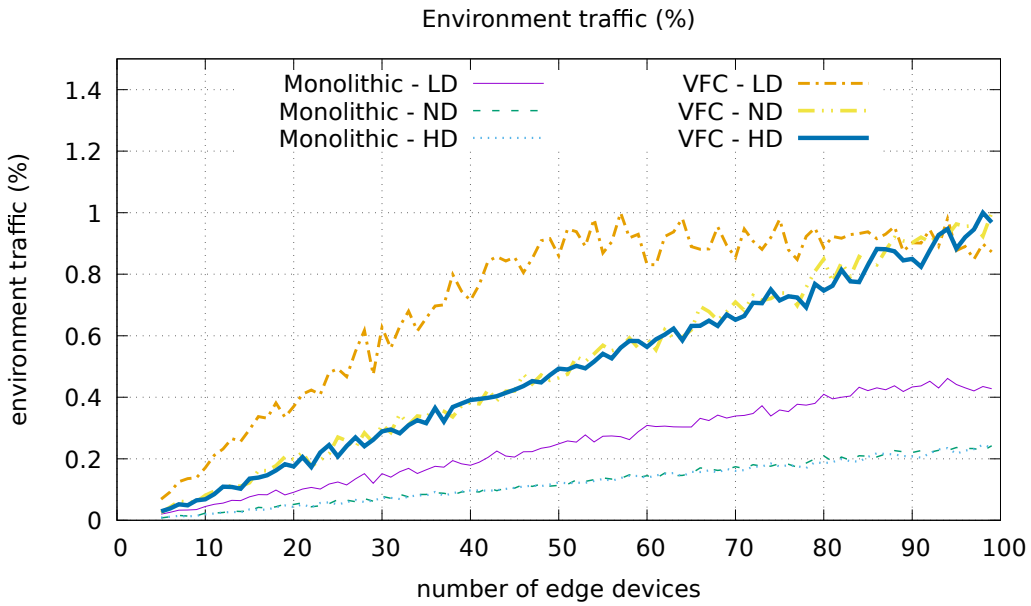

Figure 3·22: Edge environment network traffic on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices

Table 3.4: Evaluation of VFC services - single service (*Service A* (gender classification)).

Scenario	Description
S11	VFC model.
S12	VFC model + QoS mechanism.

Table 3.5: Evaluation of VFC services - multiple services (*Services A* (gender classification) and *B* (age classification) deployed on $VFCs$).

Scenario	Description
S21	VFC model.
S22	VFC model + cashing mechanism.
S23	VFC model + cashing + QoS mechanism.

3.4.2 Assessing the add-on services on the VFC model

As described in Section 3.3.2, a prototype edge environment with eight low capacity edge devices has been implemented. Using this environment, a set of experiments (scenarios) have been implemented, as presented in Table 3.4 for a single service and in Table 3.5 for multiple services.

In order to support *Service A*, VFO deployed 1 instance of $VF1$, 2 of $VF2$, 4 of $VF3$ and 1 instance of $VF4$. For *Service B*, the resulted instances were 1, 3, 3 and 1 for $VF1$, $VF2$, $VF5$ and $VF6$ respectively. The calculated $VFCs$ are in accordance with the VF characteristics, as the most demanding VFs ($VF3$ and $VF5$) participated in the chains with the largest number of instances.

QoS monitoring mechanism

This set of experiments aim at assessing the performance of the QoS mechanism based on the learning service build on the LSTM models described in Section 3.2.3. For this, scenarios described in Tables 3.4 and 3.5 have been utilized. Each scenario has been implemented in the edge benchmark environment for a real case scenario of 12 hours of continuous execution of the surveillance services.

The results of this set of experiments are reported in Fig. 3·23 and in Fig. 3·24. QoS monitoring mechanism, which is supported by the two LSTM models described in Section 3.3.3, plays a crucial role in the stabilization of the QoS of the executed services. This applies to both single-service and multi-service scenarios, as presented in both figures. Applying the QoS mechanism decreases the variation of the processed frames by more than 59.24%, resulting to a more stable surveillance service.

Caching mechanism

As described in Section 3.2.4, the VFC model enables caching the processed data and avoiding recalculations when two of more services require the processing of the same frames. A set of experiments has been conducted, trying to reveal the benefits of the VFC approach. *Services* A and B can share $VF1()$ and $VF2()$.

Fig. 3·24 presents the results on the evaluation of the caching mechanism. The different scenarios were configured in order to support the QoS constraints of *Services* A ($fps = 12$) and B ($fps = 10$). Cashing mechanism reduces the environment cost by 32.3% from the VFC, while improving the the QoS by 2.93% and the edge environment cost by 9.8%1 compared to scenario S21.

Caching mechanism reduces the total environment cost by 30% while maintaining the QoS of the services. Additionally, the utilization of the caching model reduces the fluctuations of the QoS, which is crucial for the service delivery.

3.4.3 Assessing the performance of the VFC model

This set of scenarios aim to provide evidence about the performance of the VFC model against alternative approaches. More specifically, section 3.4.3 describes the results on the comparison of the VFC model with a cloud-based surveillance service while section 3.4.3 includes the relative results on the comparison of the model with other distribution schemes.

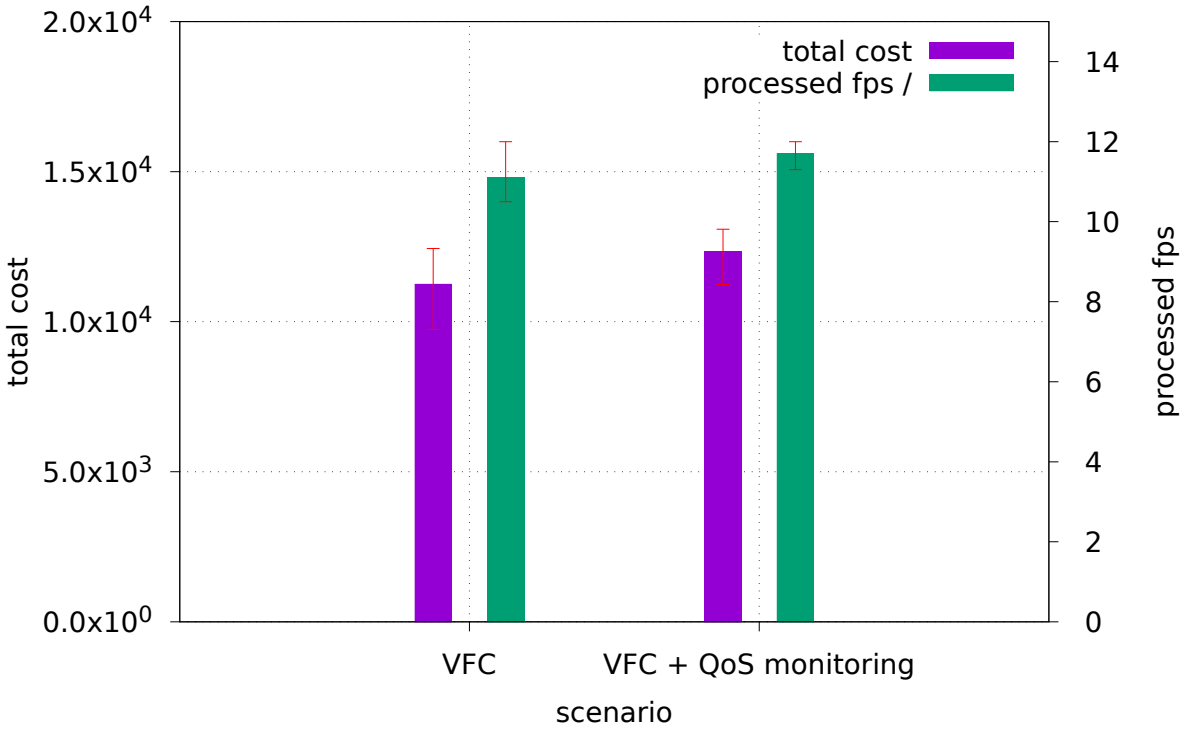

Figure 3·23: Total cost and processed fps for (a) VFC model, (b) VFC model with QoS model enabled. Single service mode.

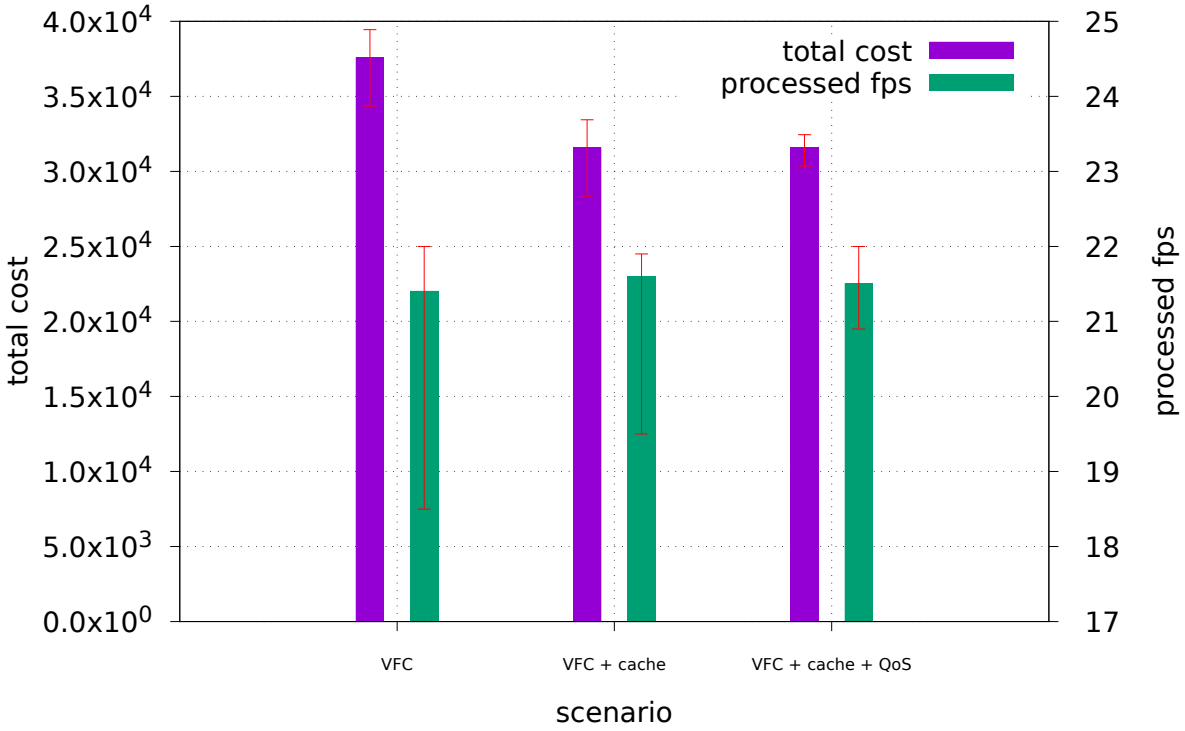

Figure 3·24: Total cost and processed fps for (a) VFC model, (b) VFC model with caching enabled and (c) VFC model with caching and QoS model enabled. Multiservice mode.

Cloud infrastructure compared to VFC model

The first set of experiments aim at evaluating the performance of the proposed VFC model against a surveillance service deployed on a cloud infrastructure. For this, *Service A* has been deployed to the cloud environment described in Section 3.3.4 and specific performance metrics have been compared against the VFC model deployed on the edge environment detailed in Section 3.3.2.

The first parameter under consideration is the performance of the cloud environment under different network communication channels and on different video resolutions. These experiments, which results are reported in Fig. 3·25, aim to assess the importance of different broadband communication technologies to the real time QoS (processed fps) of the deployed service.

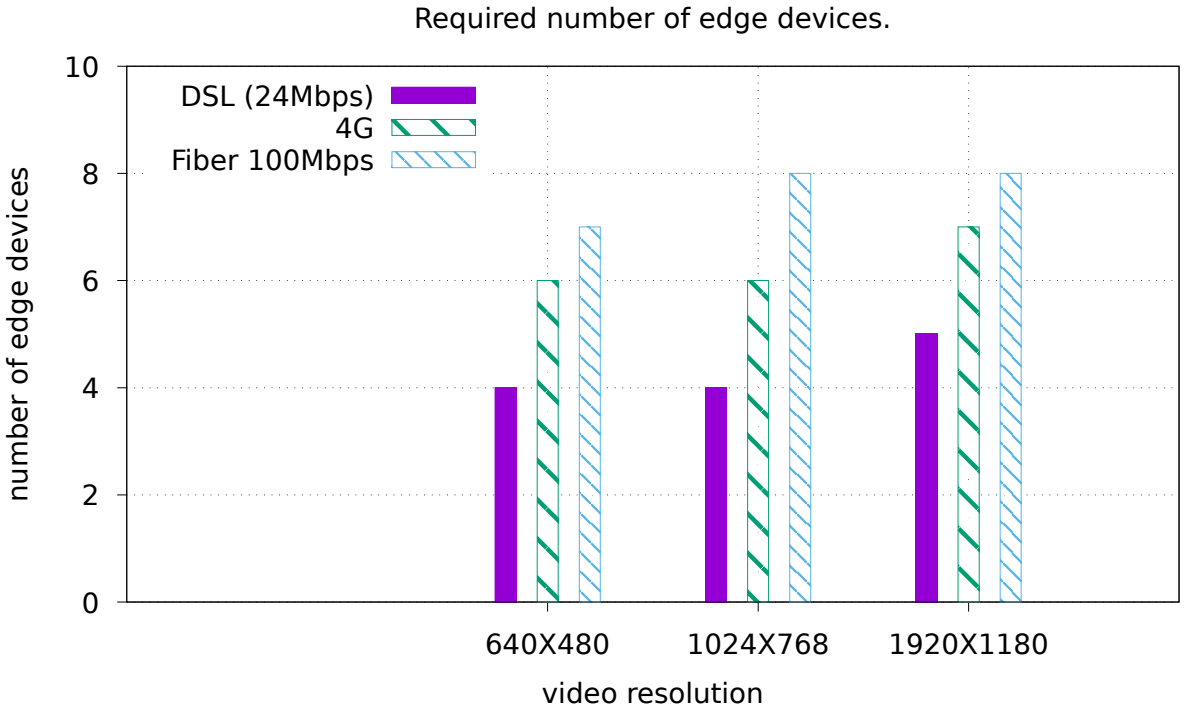

Figure 3·25: Required number of edge devices

It is obvious that the network link between the surveillance camera and the cloud

infrastructure plays an important role to the QoS of the service. Next, the second parameter under consideration are the number of edge devices required to meet the same QoS as the one observed on the cloud infrastructure. This parameter has been calculated for the different video resolutions. The results, which are presented in Fig. 3·26, indicates that with a relative small number of edge devices, VFC model can meet the performance of a cloud service.

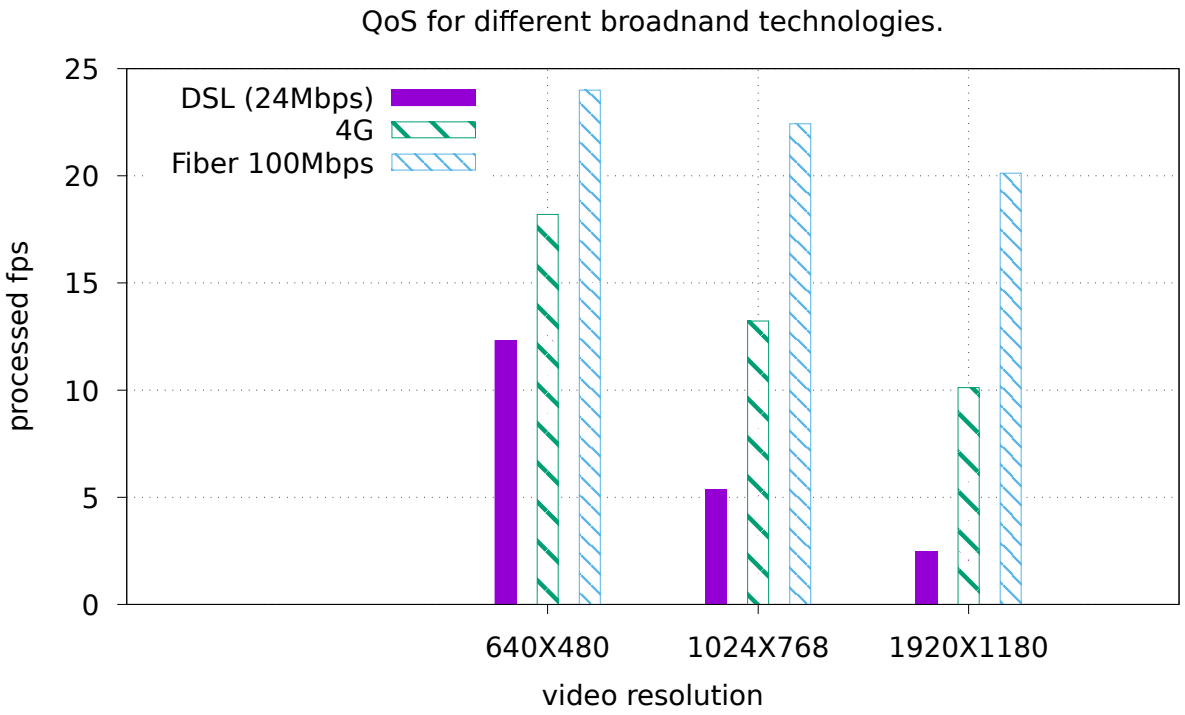

Figure 3·26: QoS for different broadband technologies

Comparison with similar distribution frameworks

The experiments for this purpose are summarized in Tables 3.6 and 3.7. More specifically, the VFC model has been tested against $Monolithic$ and $SFDM$ models, as well as with the Apache Spark ©both on single service and on mutli-service scenarios. The relative results are presented in Fig. 3·27 and in Fig. 3·28 respectively. Both for the single service and the multi-service scenarios, VFC model achieved higher

Table 3.6: Comparison of VFC model - single service (*Service A* (gender classification) against other frameworks).

Scenario	**Description**
S31	Monolithic model.
S32	SFD model.
S33	Spark framework.
S34	VFC model + QoS mechanism.

Table 3.7: Comparison of VFC model - multiple services (*Services A* (gender classification) and *B* (age classification) against other frameworks).

Scenario	**Description**
S31	Monolithic model.
S32	SFD model.
S33	Spark framework.
S34	VFC model + cashing + QoS mechanism.

QoS compared with the *Monolithic* approach (+120.5% for the single service and +133.3% for the multi-service scenario) and with the $SFDM$ model (+22.5% for the single service and +10.8% for the multi-service scenario). At the same time, the VFC model reduced the operational cost by 43.5% on average compared with the aforementioned models.

As far as the comparison with the Apache Spark © framework is concerned, the VFC model has achieved on average the same QoS by reducing the operational cost of the edge environment by approximately 103.3% on the single service scenario and by 90.7% on the multi-service scenario. Additionally, while the average value of the achieved QoS by the VFC model is slightly improved with the one produced by the Apache Spark ©, the variation of the QoS throughout the service is reduced by almost 50.3% in both single service and multi-service scenarios.

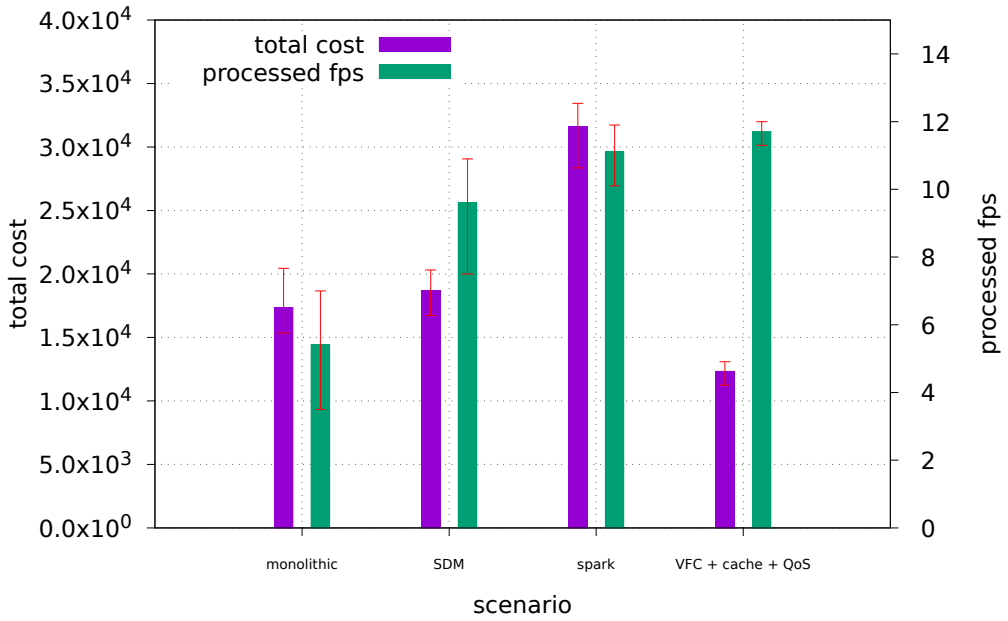

Figure 3·27: Total cost and processed fps for (a) Monolithic model, (b) SDM, (c) Apache Spark and (d) VFC model with with caching and QoS model enabled. Single service mode.

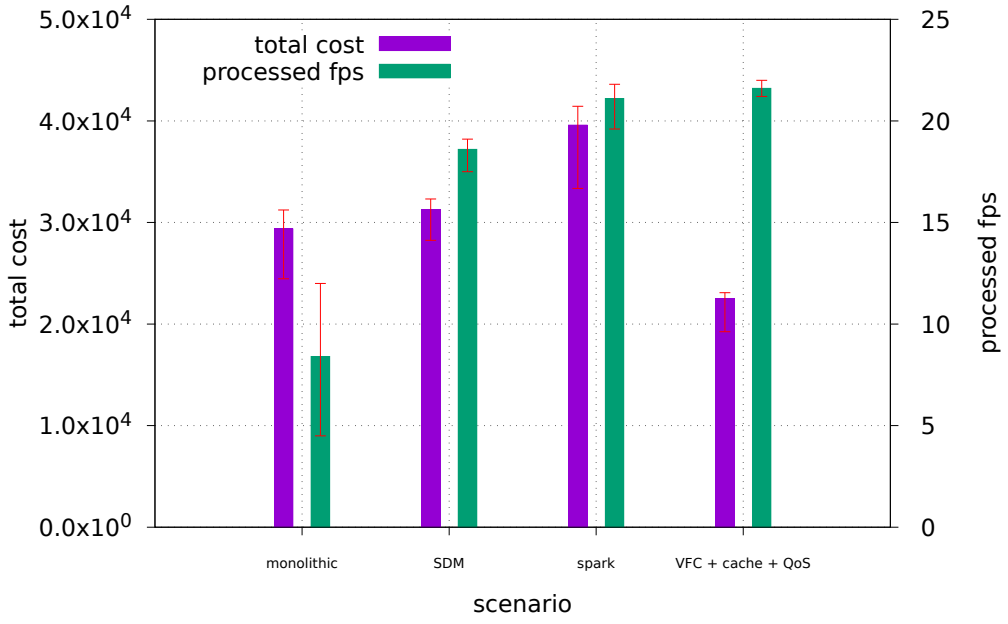

Figure 3·28: Total cost and processed fps for (a) Monolithic model, (b) SDM, (c) Apache Spark and (d) VFC model with with caching and QoS model enabled. Multi-service mode.

3.5 Discussion

Enabling edge computing to support heavy load computational tasks is crucial for integrating IoT environments with machine learning and artificial intelligence services. Towards this direction, a novel approach is proposed, which introduces the decoupling of the services to independent micro-services, all integrated under a VFC model.

The proposed VFC model is introduced by incorporating a decoupling scheme on the main VF allocation process. It is shown that this particular NP hard problem can be solved in a viable time-frame, even for edge computing low level devices.

Aiming to assess the scalability and the performance of the proposed VFC model, a comparative study has been performed, both on a simulator and on a real-case benchmark edge environment. As far as the scalability and expandability of the model is concerned, the simulation results have revealed that the VFC model can be deployed on a high number of edge devices, maintaining each advantages as far as the total cost and the edge devices utilization are concerned. Additionally, the proportions of the served services and the rejected services over the total requested services under different user demand rates show that the VFC model can operate effectively on large-scale scenarios.

As far as the comparison of the VFC model with alternative distribution approaches is concerned, a set of experiments has taken place. The experiments can be categorized into three main categories. Namely, the first category involve the comparison of the VFC model with baseline approaches, like the $Monolithic$ service approach and the $SFDM$ model. The scope of these experiments is to set a performance borderline, aiming to assess the improvement level of the VFC model. The relevant results have shown a substantial improvement over the aforementioned simplistic approaches.

The other two categories of experiments involve more pragmatic alternative ap-

proaches. Namely, a comparison study with a surveillance service deployed on a Cloud infrastructure has been held. The results have shown that with a relative small number of edge devices, the VFC model can have the same performance metrics as the Cloud service, under different technologies of broadband connections. Finally, a comparison study has been performed with Spark, a generic distribution framework. Both due to the extensive footprint of Spark on the low-level edge devices and due to the achieved QoS, the VFC model has outperformed Spark in all performed scenarios, especially on stabilization of the achieved QoS.

On top of the VFC model, two add-on services have been designed, developed and deployed, aiming to boost its performance. More specifically, a caching mechanism has been introduced, reducing the operational cost of the edge environment on multi service usage scenarios. Finally, QoS monitoring service based on a deep learning framework attempts to predict possible VFs failures and inform the VFO to take the appropriate actions.

All of the aforementioned experiments support the fact that the proposed VFC model has the capacity to effectively distribute and deploy complex service and AI models. The main aspects of the proposed model include the VF placement and consequently the setup of the edge environment, as well as a VF migration model for auto-healing purposes. Finally, the framework supports caching among different virtual function chains, boosting the performance and reducing the overall environment cost.

As far as the limitations of the proposed model are concerned, its main shortcoming is the fact that all the VFC services (placement, caching, migration) are deployed centralized at the VFO node. This produces a single-point failure discrepancy, which limits the capacity of the proposed framework, in terms of robustness. As future work, a de-centralized model, based on distributed algorithms should replace

the VFO node, according to which all involved nodes will undertake the support of the aforementioned services.

Chapter 4

VFCSIM: A simulation framework for real-time multi-service Virtual Function Chains deployment

Modelling service deployment on distributed, heterogenous and dynamic environments usually involves the integration of multiple software components under a common deployment. Such deployments have gained focus lately, not only because of the rise of the Edge Computing paradigm but also due to Clouds and micro-Clouds Computing infrastructures. A novel approach for this deployment is Virtual Function Chains, according to which a service is decomposed to a set of Virtual Functions and each Virtual Function is undertaken by a different device. We propose a Virtual Function Chain Simulation ($VFCSIM$) Framework, an opensource library for building simulation models of both edge and cloud networked systems, based on Virtual Function Chains. $VFCSIM$'s central structure is a heterogeneous network, which can describe, via scripting, a wide set of devices, network links, cloud resources, cost models and services. A case study is presented, according to which final users request on demand surveillance services from an edge network. After describing the simulation setup steps, both the summarization results and the real-time probing metrics are presented. $VFCSIM$ is expected to enhance and facilitate the deployment of the Virtual Function Chaining paradigm both to cloud and edge infrastructures.

4.1 Introduction

Engineers and researchers usually use simulation frameworks to optimize and evaluate their edge computing architectures and designs. While simulators offer many advantages, various modelling challenges need to be tackled by developers (Kecskemeti et al., 2017). Compared with mainstream cloud computing simulators, edge computing environment simulators need to incorporate heterogeneous devices, limited computing resources and networking protocols, which require a more complex framework to be developed. Thus, while cloud environment simulators can inspire and direct the device of edge based simulators, they can not be used 'off the shelf' for simulating edge environments.

A second challenge to be addressed when designing an edge environment simulator is the mobility of the edge devices. Relative positioning of the edge devices plays an important role when point-to-point communication protocols are applied (e.g., BLE – Low Energy Bluetooth). Finally, edge based setups are highly fluctuating environments where edge devices become either available or unavailable constantly. This environment fluctuation is a characteristic that is not explored in cloud based simulators, as the infrastructure is expected to be stable at a high degree.

A service deployed on an Edge environment can be conceived as a set of independent functions which communicate with each other to realize the service's objective, and typically interact in a sequential order, especially in IoT scenarios demanding for sense-process-actuate workflows (Brogi and Forti, 2017), (Skarlat et al., 2017). These functions are composed of atomic commands and are expected to be deployed on virtualized infrastructures (e.g., container-based virtualization (Morabito et al., 2017)) or run natively on an edge device. For instance, Docker containers are explored in Microsoft Azure IoT Edge (Mendu et al., 2022) for deploying computational processes on edge devices, and Amazon AWS Greengrass (Dayalan et al., 2021) can deploy and

execute Lambda functions on the Greengrass core software in edge devices (Das et al., 2018).

An umbrella term for these functions is called Virtual Functions (VFs) and are inspired by the Virtual Network Functions, as a set of interconnecting and communicating VFs as Virtual Function Chains ($VFCs$). Concerning VFC models, a wide set of problems have been addressed and discussed by the research community (Gouareb et al., 2018). Among the most challenging ones are:

1. VF **placement problem**, which refers to the optimal assignment of each VF instance to an edge device, so that the services constraints are met and the total environment cost is minimized.

2. VF **dimensionality problem**, which refers to the estimation of the minimum number of VF replicas required to implement the service. It is important to mention that VFC models may consider deploying multiple replicas of a VF in order to distribute the data to the edge and thus reducing the processing time of the specific VF. This approach, inspired by the micro-services model (Sun et al., 2017), can enable the deployment to heavy processing VFs to low-end edge devices.

3. VF **migration problem**, which refers to the transparent transfer of the running status of a VF when the undertaking device needs to be removed from the VFC. While the aforementioned problems have been discussed in the literature, new approaches appear constantly which need to be evaluated.

For this, we consider a simulation framework suitable for modeling VFC architectures and schemes that would be of great usefulness to engineers and researchers. This work proposes $VFCSIM$, a simulator for VFC architectures, both in the Edge and in Cloud. While other simulation environments can be used for simulating $VFCs$,

they require substantial effort for doing do.

The novelty of $VFCSIM$ is that is can natively support VF and $VFCs$. Most important, $VFCSIM$ can support real-time interaction with the simulations, enabling the ad hoc alteration of the edge environment, edge network and deployed services. $VFCSIM$ can be used for simulations on Edge environments, as well as in Cloud environments. Finally, $VFCSIM$ can support multi user access support, which is crucial for edge environments. $VFCSIM$ is an open source project and is publicly available at ***github.com/vtsakan/VFCSIM***.

$VFCSIM$ introduces a simulation framework which can be used to simulate scenarios based on virtual functions, both on edge and cloud environments. The main novel aspects of the proposed framework are: (i) native support of VF and $VFCs$, (ii) real-time interaction with the simulation environment, (iii) common simulation environment for Edge and Cloud based simulations and (iv) multi user access support.

4.2 $VFCSIM$ simulation framework

With the progress of modeling frameworks, and the diverse scenarios where VFC could be integrated, the scripting approach has been applied, according to which the modeler can utilize already established entities and constitute a simulation scenario, without requiring advance programming skills, especially with Python involved in the pathway (Thorp and Bronson, 2013). When a flexible structured framework is provided, scripting can create complex simulation scenarios by modelers, after only a small time investment which is required for catching up with the learning curve.

$VFCSIM$ is a modelling and simulation framework created to build VFC scenarios on edge and cloud infrastructures. $VFCSIM$ is based on an object-oriented approach to define the network environment, the computational nodes (edge-cloud nodes), the services, described as a set of sequential VFs their properties, as well as

the demand from the users side. This particular design approach is general enough for enabling modular, multi-services scenarios and specific enough, so that the modeler can easily deploy different scenarios.

$VFCSIM$'s novelty lies on the fact that it is a generic VFC simulation framework, written in Python, capable of supporting multi-service modelling. The use of Python offers $VFCSIM$ particular advantages, such as:

- Python integrates numerical, statistical, scientific and visualization libraries, like numpy, pandas, seaborn, matlibplot and skipy-kit.
- Python is a recognized framework for resource modelling (Fienen and Plant, 2015)
- Python integrates pyomo, a native optimization framework.

In $VFCSIM$ documentation, the following terminology is used to describe its elements:

- *Module*: A file containing a python class definition,
- *Object*: An instance of a class (as defined within a Module),
- *Type*: A file containing a super-class definition, along with the relative python decorators and
- *Simulation*: An instance of a model with a particular set of model parameters.

4.2.1 Design

The basic classes of the simulation framework are presented in Fig. 4·1. More specifically, the basic entities of the framework are:

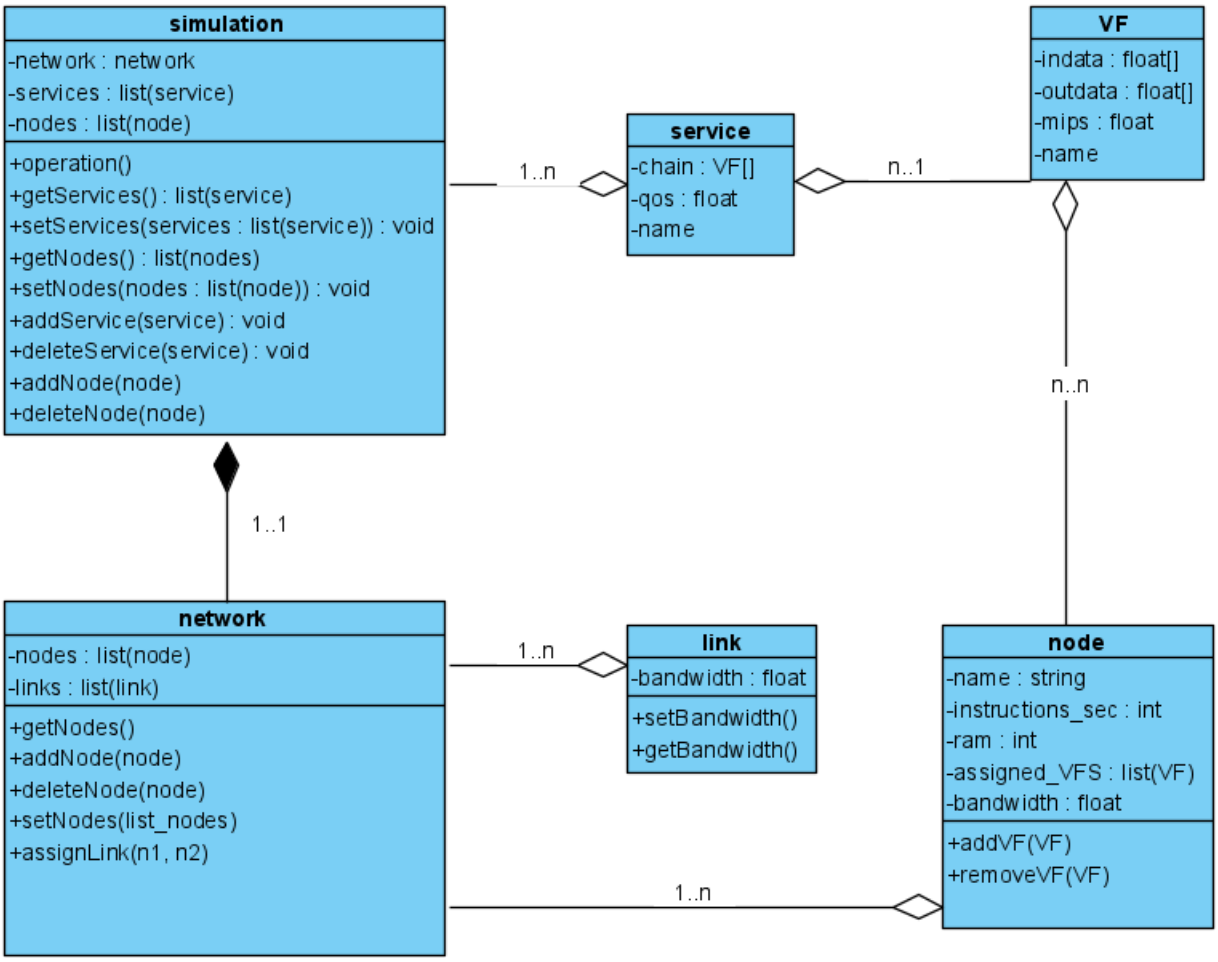

Figure 4·1: UML design of the proposed simulation framework.

- ***Node***, which is used to model a computational and communication entity, as an edge device or a virtual machine. The basic attributes of the Node entity are (i) CPU mips (million instructions/sec), RAM memory volume, storage capacity and a cost function, which refers to the total operational cost of the node

- ***VF***, which is used to model a virtual function with attributes: (i) input data size, (ii) output size data and (iii) required CPU instructions

- ***NetworkLink***, which is used to model a point-to-point or a point-to-multipoint network communication links.

- ***Service***, which models a deployed service. Service's attributes include a set (list) of VFs, which actually constitute a VFC and QoS requirements which refers to the specific quality of service contract of a service.

- ***Simulation***, which is used to create a script with a specific simulation scenario.

In order to set up a simulation scenario, one should create a script which describes (i) the device environment, (ii) the deployed services and the networking infrastructure. Each simulated entity is materialized as a Python process, utilizing the multiprocess Python library. Thus, $VFCSIM$ considers a no-shared memory environment, promoting a realistic framework for the simulations. $VFCSIM$ handles the evolution of time as discrete time points. It is important to mention that the time discretization is involved on probing the relative pre-defined metrics and logging them. The actual simulation of performed on a realistic basis, according to the specific simulation scenario. Fig. 4·2 illustrates the basic steps for the execution of a simulation.

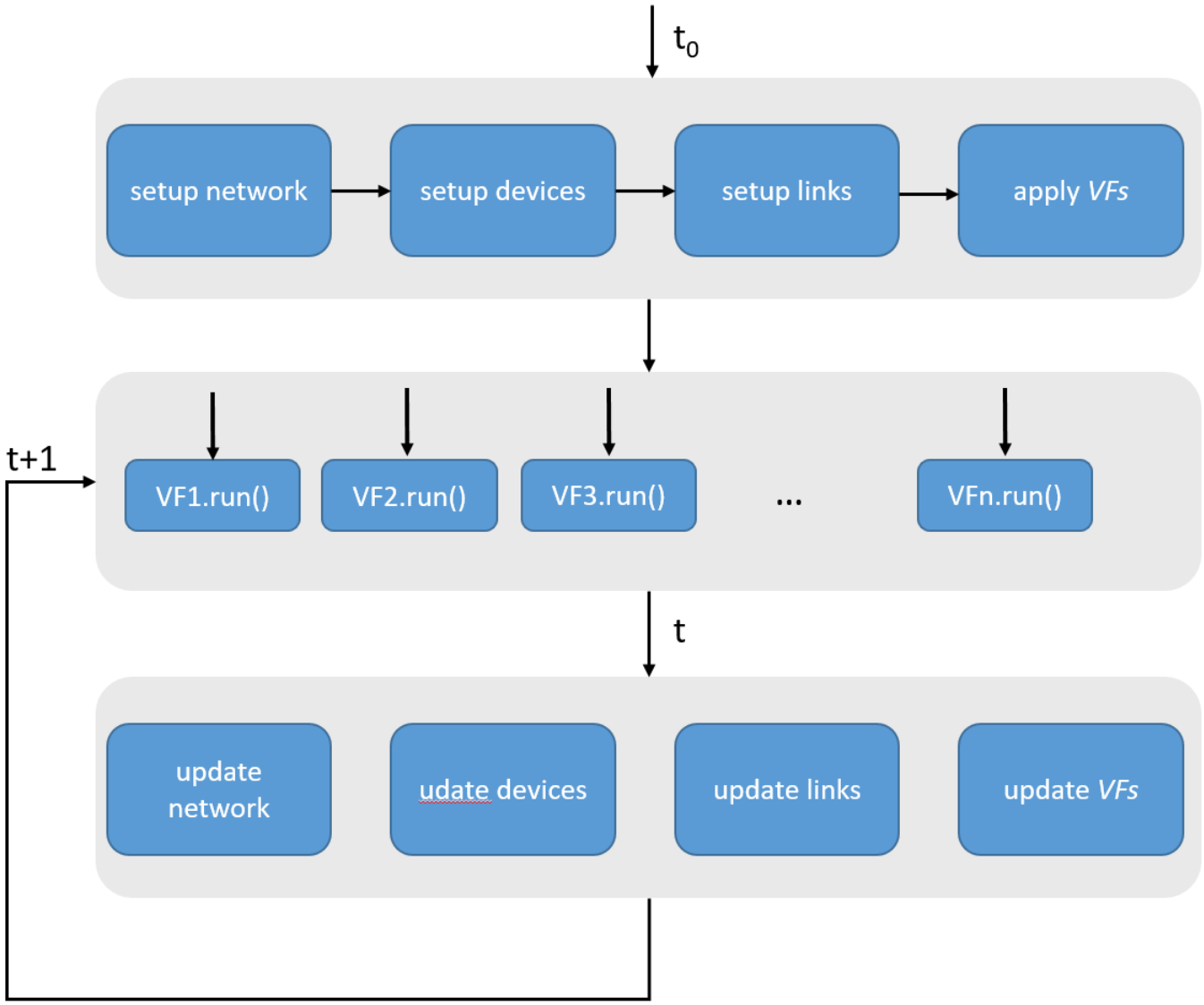

Figure 4·2: Steps for simulation execution.

4.2.2 Visualization

VFCSIM offers two visualization modules, which are expected to facilitate the work of the modelers. The first module parses the designed environment and produces a graph network by utilizing the NetworkX library (Prabhakar and Anbarasi, 2021). Thus, the modeler can inspect the designed network. It is important to mention that $VFCSIM$ offers live interaction with the simulation, which means that the modeler can intervene with the environment during the execution of a simulation (Fig. 4·3). The second visualization module refers to the real-time probing of specific simulation metrics. This user interface explores the Python-Qt library and communicates with the up-running processes via UDP multicast sockets. Thus, when an edge node for example needs to report a metric, it creates a multicast packet and send to a pre-defined port. The visualization module dynamically creates a set of views for each node and for each VF, which can be later selected by the modeler (Fig. 4·10).

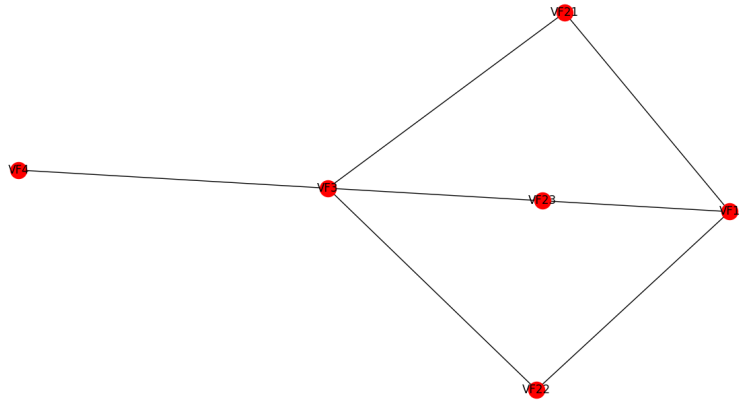

Figure 4·3: Network visualization of the simulated environments.

4.3 Validation

Aiming to verify that the entities built in $VFCSIM$ are valid, and that the simulation environment is functioning properly, we have implemented a virtual function chaining scenario in $VFCSIM$ and in a well-established simulation environment (iFogSim). The scenario involved the modeling of a set of virtual function chains, each one of them modeling a mock service. Each VFC required 3 - 8 nodes, with a different VF to be deployed to a different edge node. Each VF produced a throughput of 200kbps. By implementing the same setup in both iFogSim and $VFCSIM$, the overall network utilization has been compared (Fig. 4·4). One can observe that the difference between the two simulation frameworks, is approximately 1.8% (*maximum value*), and remains relatively small, even when the number of the edge devices increases.

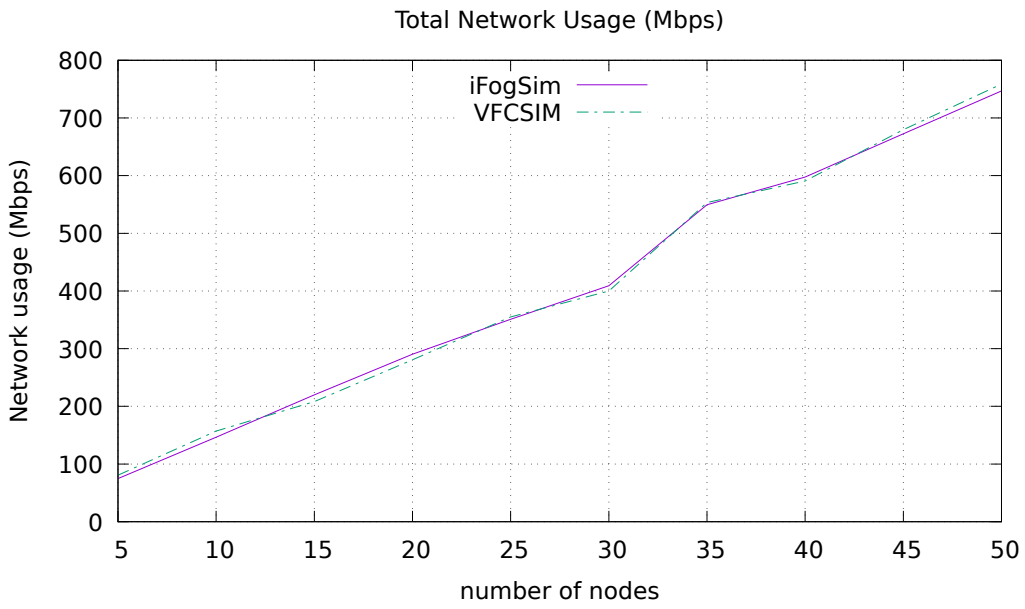

Figure 4·4: Network utilization on $VFCSIM$ and iFogSim.

4.4 Case Study: Surveillance services

Aiming to test the proposed simulation framework, a specific scenario has been built. More specifically, an edge environment with dynamic number of devices has been considered, upon which, users can request on demand different surveillance services with different demand probabilities. The simulated services have been modeled as a set of n VFs, where n is a random integer $\in [4, 10]$, following a normal distribution. Each VF could be either a light VF, a moderate VF or a heavy VF, with relative computational characteristics each. Withing the simulation platform, two different Setups of a surveillance service have been implemented.

- **Setup I**: The surveillance service has been implemented under a monolithic approach. This means that all VFs of the same service has been hosted in one edge device.
- **Setup II**: A VFC model, where the different VFs were deployed on different edge devices.

The two aforementioned setups have been tested under different service demand probability distributions. By service demand probability distribution, we refer to the probability a user requests a service at a specific time-point t_d. More specifically, various Poisson distributions have been used. For the Poisson distributions, three different λ parameters have been used, aiming to simulate low, normal and high user demand rates. For the same setup, the cumulative number of the requested services is the same.

The parameters of the simulation environment are given in Table 4.1.

For this specific scenario, the modeler aims at exploring the scalability of VFC model. For this, a script has been developed which implements the aforementioned setups under two dependent variables: (i) the number of edge devices n and (ii) the

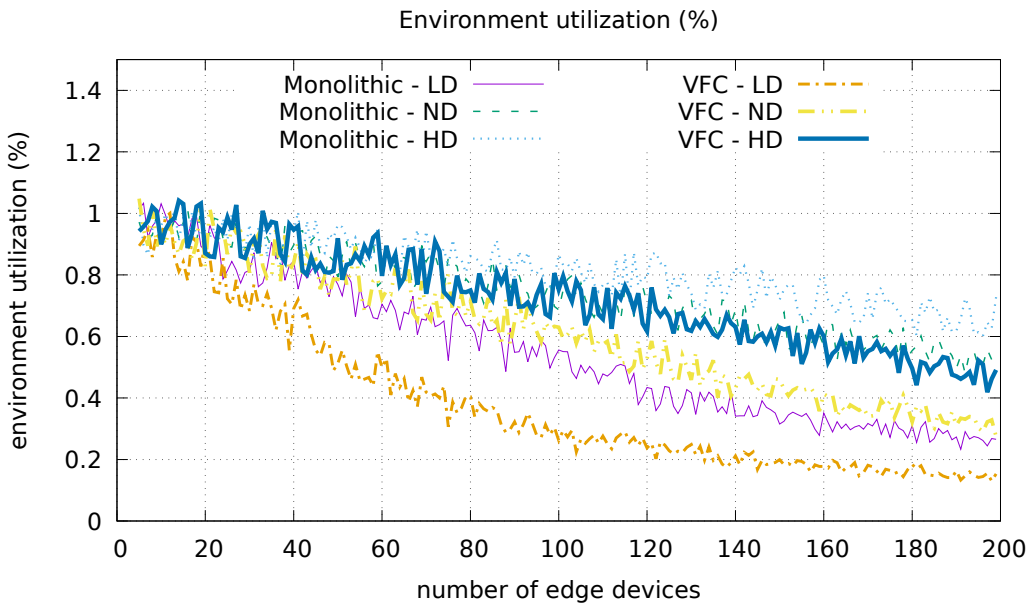

Figure 4·5: Edge environment utilization on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

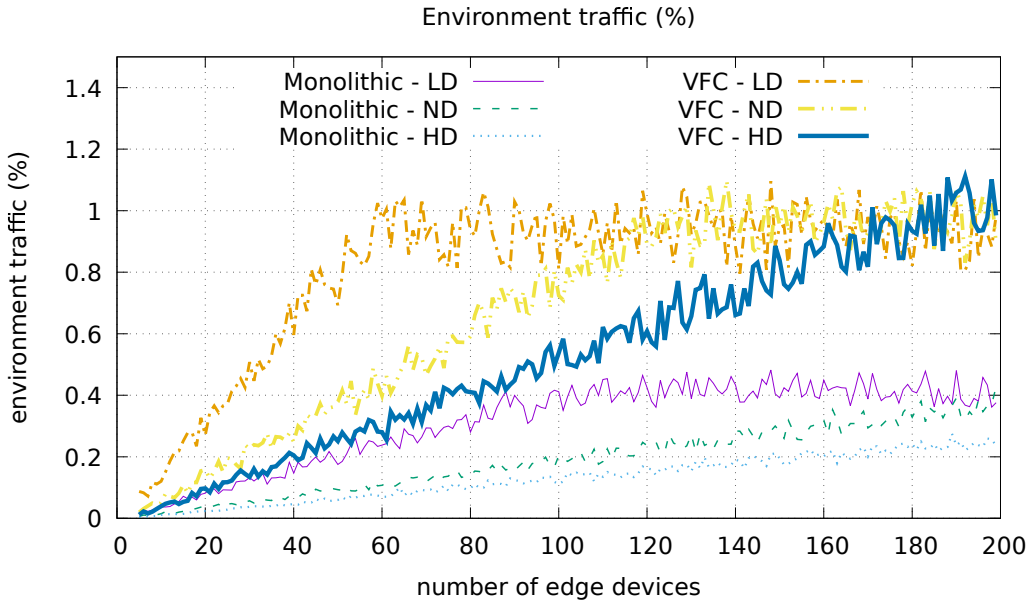

Figure 4·6: Edge environment network traffic on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices

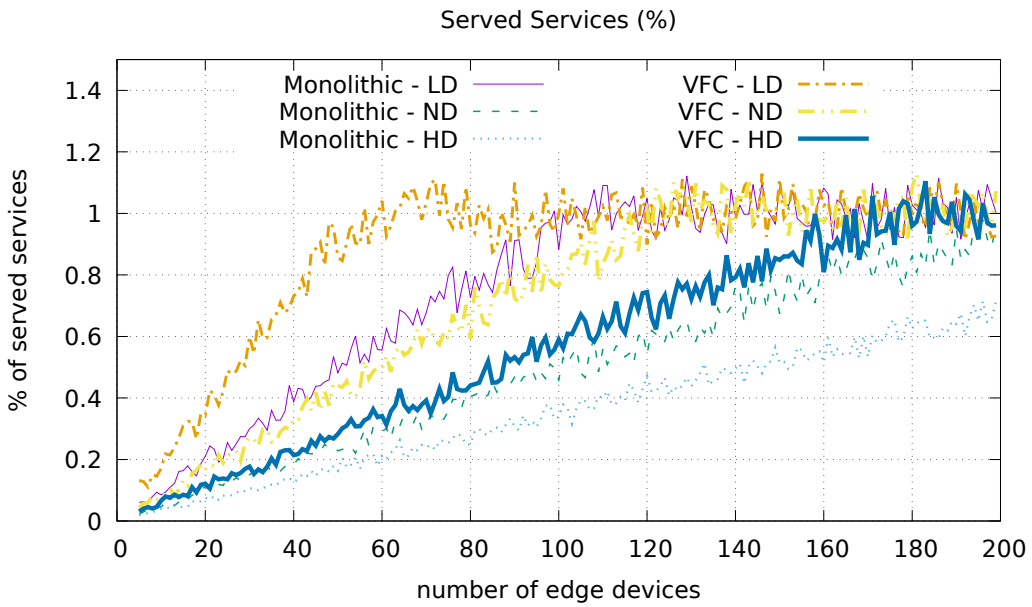

Figure 4·7: Percentage of served services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

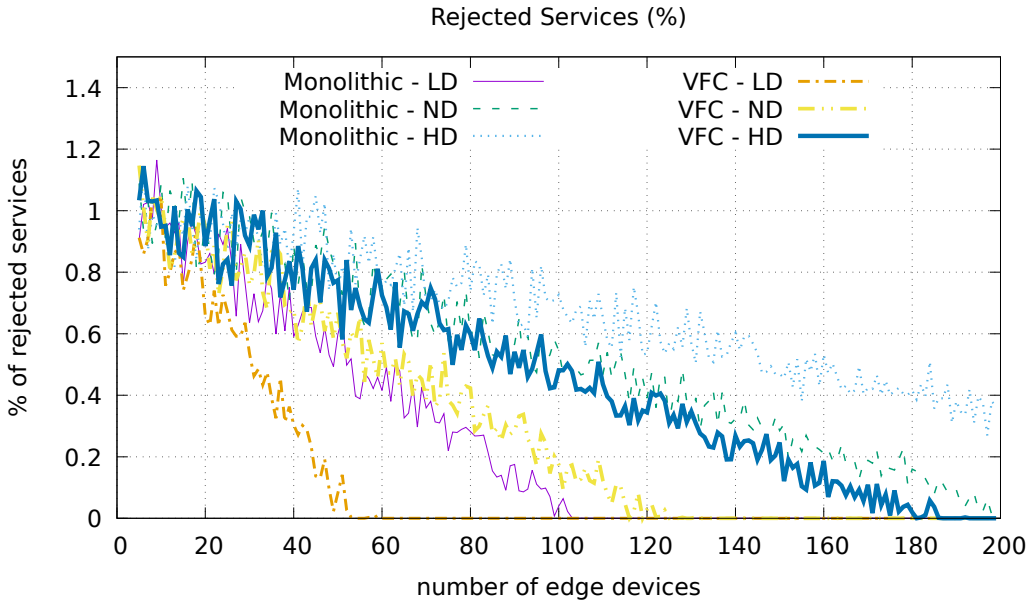

Figure 4·8: Percentage of rejected services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

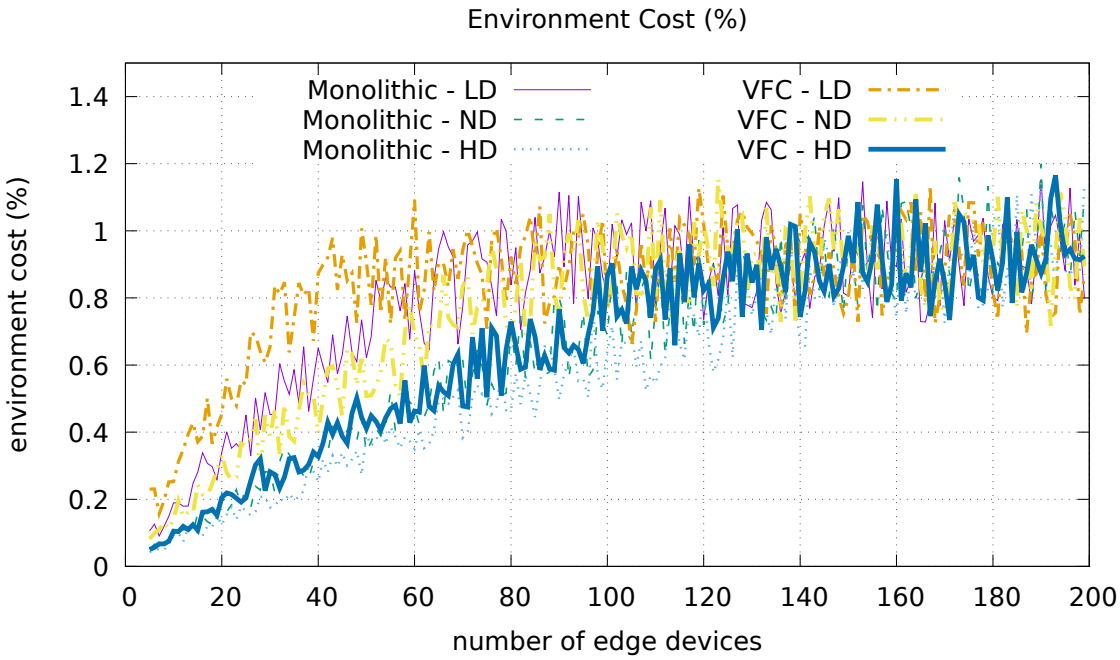

Figure 4·9: Edge environment cost on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

Table 4.1: Parameters on simulation setup.

Parameter	Value	Description
cpu_{load}	$10^4 N(12, 0.9)$	cpu mips of edge devices
m_k	$2 \times 10^3 N(30, 0.7)$	VF_k required mips per data instance
Wtt'	$3 \times 10^5 N(13, 0.9)$	bps between VF_t and VF'_t
$c_g(l)$	$\frac{l^2+1.2}{1400}$	function which describes the cost of g edge device when performing l instructions
$r(l)$	$\frac{30l+11}{m_k}$	function which describes required time of an edge device to perform l instructions

service demand distribution probability. The measurements of the simulation study include the following metrics:

1. the total edge environment utilization (Fig. 4·5),
2. the total network traffic (Fig. 4·6),
3. the percentage of successfully served services (Fig. 4·7),
4. the percentage of rejected services (Fig. 4·8) (inadequate resources) and
5. the total edge environment cost (normalized) (Fig. 4·9)

. Finally, the $VFCSIM$ visualization module can probe specific metrics during the execution of the simulation. For example, in Fig. 4·10, the available RAM of edge device 2 is presented, after 60 secs of simulation time.

4.5 Conclusions

$VFCSIM$ has been applied successfully to a case study which aims to design an edge network with the capacity to undertake a set of surveillance services. The feature under study was the number of required edge devices against the service deployment mode (Monolithic vs. VFC). Via scripting, a set of simulation scenarios

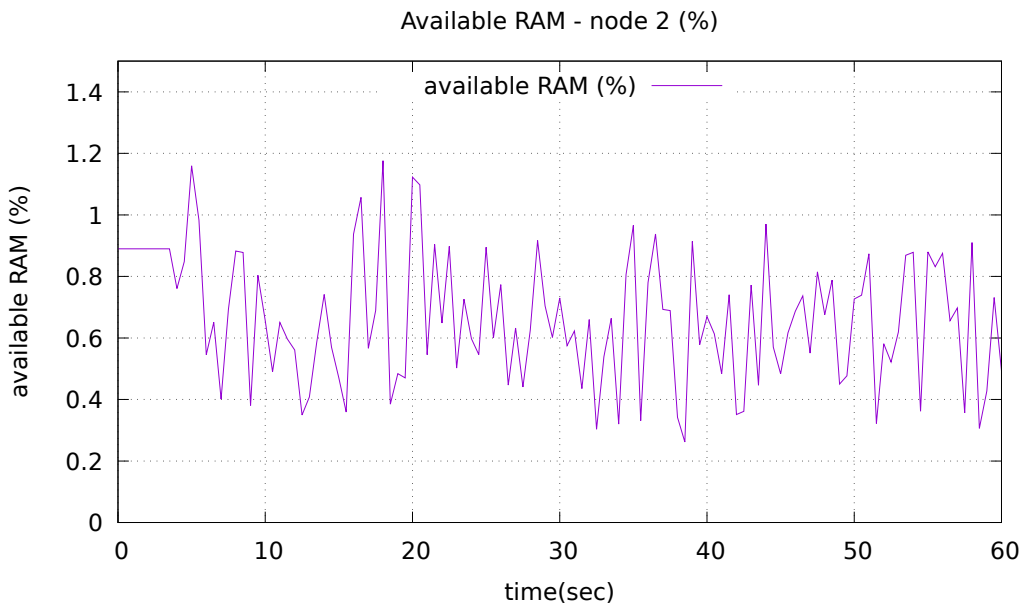

Figure 4·10: Real time graph - Available RAM of NODE 2.

have been considered, with different end user demand probability distributions. From the simulation results, certain conclusions can be drawn. For instance, in order to avoid down-time of the designed system (down-time refers to the time the system can not undertake new services), the edge environment would require 58 devices for low demand, 125 devices for normal demand and about 190 devices for high demand, if the VFC model is applied. In case of the Monolithic model, the relative numbers are 110, 195 and > 200 edge devices. Thus, $VFCSIM$ can be successfully utilized in order to design heterogeneous edge networks with the capacity to undertake streaming services, like surveillance analytics, with a certain QoS.

As virtualization and dockerization technologies are recently considered not only to the cloud but also to the edge computing paradigm, an abstract approach for developing services is required. VFC model can support this design approach. For this, we propose a simulation framework which can support modellers, engineers and researchers to test architectures and solutions based on VF. The limitations of the

framework summarize our future work. An integrated user interface for designing networks, devices and VF is expected to facilitate the work of the modellers. On the same direction, the integration of libraries with well-established devices and communication protocols will result to a more concrete simulation framework.

Chapter 5

Virtual Function Migration Model

5.1 Introduction

Compared to cloud environments, systems deployed on edge networks present core differences, especially when it comes to virtualization and migration models. In cloud infrastructures, virtualization of a service via a virtual machine or a container may facilitate isolation and flexibility, which permits occupancy and means efficiency. In such an environment, migrating a service among processing nodes offers the system suppleness and adaptability.

When compared with an Edge environment, virtualization and migration appear specific differences, due to the following remarks:

- Edge network comprises nodes which appear a high degree of heterogeneity in terms of software capabilities (e.g., operating systems) and hardware (e.g., CPU architectures). Consequently, there is the need for more generic virtualization models.

- Edge nodes have (in most of the cases) less processing power when compared with those in a Cloud environment. Thus, virtualization and migration models with smaller footprint and less overhead would be more appropriate.

- A Cloud Data Center relies on a high-bandwidth and low latency network. On the contrary, edge nodes are interconnected through a WAN and hence usually experience disconnections (Ha et al., 2017). Having this in mind, it is essential

that all the migration models for edge environments should keep the volume of the transmitted data as low as possible.

- Aggregate migration time (which equals the downtime of the service) is not considered of high importance on Cloud environments. Yet, at the edge of the network, QoS fluctuations appear throughout the whole migration process.

- edge services usually execute momentary data analysis and are thus not supposed to write to any persistent memory (e.g., the disk). On the other hand, Cloud services exploit persistent memory in a much higher degree. Therefore, migration models on the Edge could neglect it is any data written by the service to persistent memories when migrating from one edge node to another.

5.2 State of the Art

Migration at the edge of the network is one of the most challenging research topics in the area of edge computing. As edge environments are highly dynamic and comprise heterogeneous, low-end devices, the deployment of efficient migration services is mandatory for enhancing their robustness and their reliability.

Since the emerge of edge computing, there have been numerous novel approaches for tackling the problem of migration. One may categorize the proposed migration algorithms and models in two sets, cloud-edge migration algorithms (which are also referred as *offloading* algorithms) and edge-edge migration algorithms (Fig. 5·1). Additionally, the proposed model can either utilize deterministic or machine learning models.

One of the first approaches were (Rodrigues et al., 2016). Within this work, authors presented a method for minimizing service delay in a migration scenario between two cloudlet servers, after considering both computation and communication

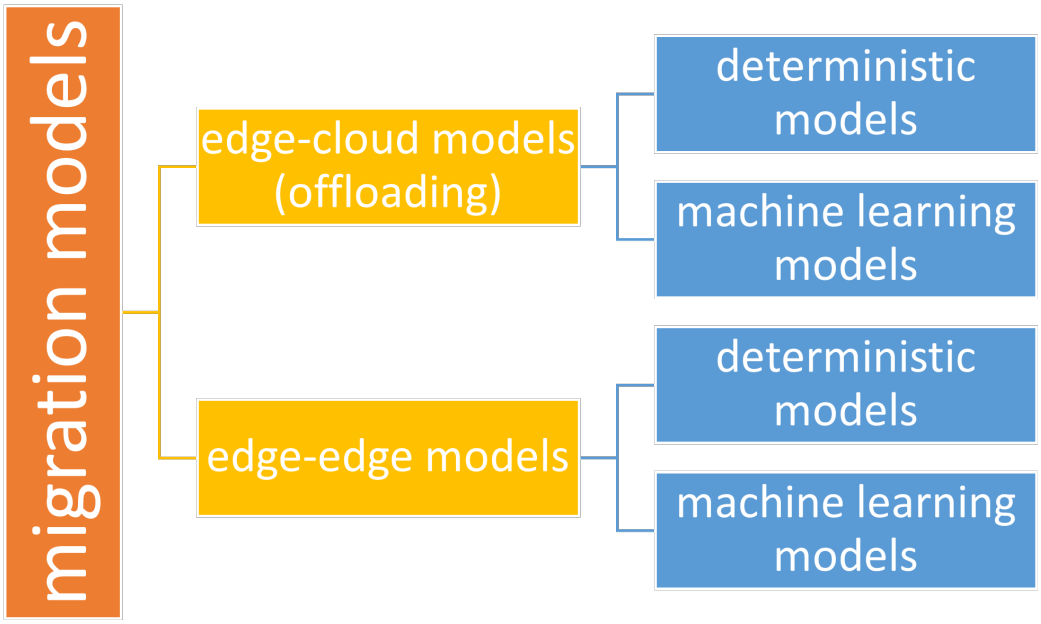

Figure 5·1: Categories of migration models.

elements, controlling processing delay through virtual machine migration and improving transmission delay with transmission power control. According to the main outcome of this work, the consideration of both computation and communication constrains results to the optimum design of a migration model. A replica migration model is proposed in (Li et al., 2019) for facilitating access hotspots to obtain the pairing migration relationship from source node to target node. The experimental results revealed that the proposed replica migration algorithm can effectively reduce the migration time, minimize the response time, and improve network resource utilization.

On the same topic, the work presented in (Doan et al., 2020) introduces a model for maintaining a consistent state of mobile-edge computing application. According to this model, state storage component is decoupled from the computing one. A key-value storage layer is proposed, to synchronize states between mobile-edge computing

servers. Subsequently, a distributed key-value store framework is proposed, which decouples mobile-edge computing application design into processing and state, to ensure service continuity. Evaluation results show that the proposed solution reduces downtime by half in most of the cases, even under high load of state update. Furthermore, under moderate load of state updates, the framework can eliminate downtime completely.

Authors in (Chen et al., 2019), after introducing edge cognitive computing paradigm, they describe an edge cognitive computing based dynamic service migration mechanism to provide insight into how cognitive computing is combined with edge computing. The experimental results show that the proposed architecture has ultra-low latency and a high user experience, while providing better service to the user, saving computing resources, and achieving a high energy efficiency. Additionally, the work presented in (He et al., 2021) models the intra-edge migration problem as a dynamic resource dependency graph. After introducing an iterative Maximal Independent Set-based multiple migration planning and scheduling algorithm, based on real-world mobility traces of taxis and telecom base station coordinates, authors provide evidence that the proposed model can efficiently schedule multiple live container migrations in large-scale edge computing environments.

As far as services migration between different edge environments is concerned, (Doan et al., 2021) introduced Flexible and Low-Latency State Transfer in Mobile Edge Computing model, a novel programmable state forwarding framework. The proposed model flexibly and directly forwards states between source instance and destination instance based on Software-Defined Networking. Mobility of edge nodes is discussed in (Ray et al., 2020), where it is discussed that edge network design and services placement may need to be re-calibrated, triggering service migrations to maintain the advantages offered by mobile-edge computing. In this work, authors proposed

a Reinforcement Learning based proactive mechanism for microservice placement and migration. Experiments on the San Francisco Taxi dataset showed the effectiveness of the proposed model in comparison to other state-of-the-art methods.

Recently, researchers attempted to explore the potential of machine learning paradigm on edge migration models. For instance, works like (Sun et al., 2018) and (Ray and Banerjee, 2021) propose a reinforcement learning based model for identifying the optimum policy for service migration. According to the later work, the migration problem is formulated as a sequential decision making problem aiming to minimize the overall response time. Then, a novel on-policy reinforcement learning based computation migration scheme, which learns on-the-fly the optimal policy of the dynamic environment is proposed. Numerical results demonstrate that the proposed scheme can adapt to the uncertain and changing environment, and guarantee low computing latency. Similarly, authors in (Miao et al., 2022) proposed a user classification mechanism based on users' mobility patterns to reduce the complexity of decision-making. Then the service migration is formulated as a Markov decision process and then a reinforcement learning-based framework is introduced, to make service migration decisions in real time in the dynamic MEC environment. Extensive data-driven experiments demonstrate the efficacy of the proposed model in reducing the system average delay. Finally, authors in (Wang et al., 2019b) suggest a formulation of the service migration problem as a Markov decision process. For tacking the underlying optimization problem, a new algorithm is proposed and a numerical technique for computing the optimal solution, which is significantly faster than traditional methods based on the standard value or policy iteration. The applicability of the proposed model is illustrated in practical scenarios where many theoretical assumptions are relaxed. The evaluations based on real-world mobility traces of San Francisco taxis show the superior performance of the proposed solution compared to baseline solu-

tions.

Regarding the cloud-edge algorithms are concerned, authors in (Wang et al., 2019a) provide a detailed survey of the specific area, sketching the overall area and research directions. In more details, authors in (Yousafzai et al., 2019) proposed a lightweight process migration-based computational offloading framework for IoT-Supported Mobile Edge/Cloud Computing. Compared with similar approaches, the proposed framework does not require application binaries at edge servers and thus seamlessly migrates native applications. Experimental work revealed that the proposed framework shows profound potential for resource-intensive IoT application processing in Mobile Edge/Cloud Computing.

5.3 Migration scenarios

At this point, we should detail the cases that the proposed virtual function migration model considers.

5.3.1 Failure – Stateless migration

By failure, we consider an edge node unexpectedly failing and disconnecting from the virtual chain which materializes a service. As it will be discussed in the next sections, migration due to failures results to the re-deployment of the virtual function to the most suitable available edge node. We are referring to such migrations as stateless migrations.

5.3.2 Overload – Stateful migration

Due to the low capacity of the edge devices, in parallel with the increasing demand of edge services, it is not unusual for an edge network to be overloaded. In such cases, migrating virtual functions to available edge nodes can result in an overall performance improvement of the hosted services.

5.4 Edge network and virtualization

Although virtualization is a long-established technology for Cloud Computing services, its adaptation for the edge networks is a relatively new field (Mansouri and Babar, 2021). From a technical point of view, virtualization may actually refers to any compute virtualization approach, as long as there is a model to abstract the runtime from the main environment (e.g., firmware, hardware) where the business logic of an application is designed to run. If we consider this definition, virtualization for edge devices may comprise several different paradigms, covering a path from virtual machines to lightweight containers.

Typical (complete) virtualization is the first version of this technology, and it has been used widely on (full-stack) guest environments. On the other hand, para-virtualization provides improvements in the form of cooperation between the guest environment and the hypervisor. According to para-virtualization a totally inverted approach is applied, based on the Unikernels (Chen and Zhou, 2021). This change points to lightweight, application specific machine images to be run directly on a hypervisor (i.e., the VM runtime) or even on bare metal. Anykernels (Tazaki et al., 2021) are a term for a modular model to the building of a Unikernel, by making OS-specific (e.g., NetBSD-based) drivers in the form of libraries. Anykernels provide improved security, lighter footprint, and faster boot times, when compared to typical VMs.

On the other hand, containers represent a more recent virtualization paradigm, which can be characterized from the low overhead deployment and lightweight execution of applications. Indeed, containers isolate only the user space environment, leaving the hardware abstraction layer as well as the application for process sandboxing to the shared kernel co-hosting them. Also, for containerization, there are a set of approaches a model could adopt, from system containers (Pérez Castillo, 2021)

(e.g., LXC, LXD), where the user-level environment mimics a full-featured operating system, to application ones (e.g., Docker), where a container is usually expected to host a single user process. Application-based containers, generally present several features (e.g., lightweight footprint) which make them highly appropriate for the network edge.

One step after containers would be to consider bare-bones approaches. Such virtualization models are called name-spacing. Namespaces are enabling technologies for containerization, but name-spacing can be much more fine-grained and limited to very specific subsets of resources. Process name-spacing can be regard as the most intense example: resource partition in that case takes place only at the process level, such that process identifiers are unique within a namespace.

Among VMs and containers, highly lightweight VMs have been proposed. An example of this approach is the Kata Containers (Randazzo and Tinnirello, 2019), which are managed by the OpenStack Foundation (Lima et al., 2019) and merge technologies from Intel Clear Containers (Arnautov et al., 2016) and Hyper runV (Debab and Hidouci, 2020). Specifically, their business logic is providing a user understanding very similar to that of containers, while guaranteeing the security and isolation advantages of a VM.

5.5 Migration models

As discussed in the aforementioned sections, a virtual function migration can be either stateless or stateful.

5.5.1 Stateless migration

Stateless migration result in a redeployment of a stateless virtual function on the receiving node. When compared to stateful migration, stateless migration models

tend to be simpler and more straightforward, as it is applied on failure nodes, which are more frequent on edge networks.

5.5.2 Stateful migration

In this section, an overview of the existing models for stateful migration of a virtual function is provided, discussing their implications on an edge network.

Hard (cold) migration

This type of models often refer either as hard (usually on edge networks) or cold migrations. This notation arises from the fact that the virtual function stops / freezes at a certain point before starting the migration process. The migrated virtual function will resume as soon as all of its state becomes available on the receiving edge node. The duration which the virtual function is not running is known as downtime.

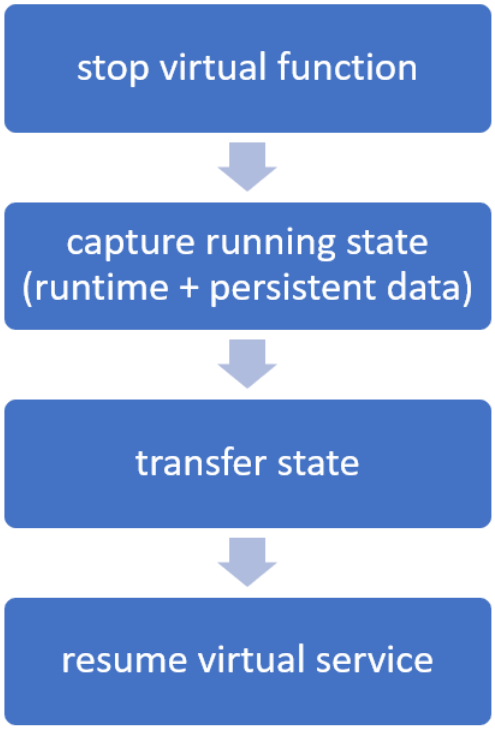

Figure 5·2: Hard (cold) migration model.

Figure 5·2 describes the necessary steps for performing a hard (cold) migration. This model is considered one of the lowest footprinted, in terms of computational requirements, which is crucial at the edge networks. However, its main drawback is the high downtime, as it even overlaps with the total migration time (i.e., the time required for the whole service state to be available on the receiving node). Finally, this model migrates the whole state (runtime and persistent data) irrespective of whether part of that state is already present on the receiving node or not. In parallel, each memory page (and disk block for persistent data) is transferred only once. A widely used implementation of hard/cold migration model is based on Checkpoint/Restore In Userspace (CRIU). While this technology has built under the Virtuozzo project (Rosen, 2016) for its OpenVZ containers (Babu et al., 2014), it quickly gained popularity and has been reused by other containerization platforms such Docker.

Soft (live) migration

Soft (live) migration models aim at minimizing virtual function downtime. Notation live refers to the fact that the virtual function keeps running while its state is being transferred to the receiving node. The virtual function is freezed only for the transmission of a slight amount of data, after which the virtual function runs on the receiving node. When the downtime is not noticeable on the QoS metrics (or by the end user), live migration is said to be *"seamless"*. Live migrations can be categorized in two categories: pre-copy and post-copy.

Pre-copy migration models (Figure 5·3)took their name from the fact that they copy the largest proportion of the state prior freezing the virtual function, after which the virtual function runs on the receiving node. It is also known as "iterative migration", since it performs the pre-copy phase through several iterations. Each iteration updates the target node with the latest state. The pre-copy phase ends when a maximum number of iterations is reached (usually predefined) and/or when

the last iteration was so short that the number of dirty pages to be transferred would determine a short downtime. The downtime of pre-copy migration mode is usually short. Yet, it can not be pre-calculated, as it as it depends on the number of dirty pages (memory pages with new data) that have to be transferred while the virtual function is suspended. This fact can cause discrepancies on the overall QoS, and need to be treated with caution. Therefore, this model is usually deployed when virtual functions have a low page modification rate. Pre-copy migration is by far more widely used than post-copy migration models. All hypervisors (e.g., VMware, Microsoft Hyper-V) implement live migration. As far as containers are concerned, CRIU provides all the basic mechanisms (e.g., CRIU pre-dump functionality) that are necessary to pre-copy the runtime state of a service.

Post-copy migration models operate on a reverse rationale, compared with pre-copying. These models initially suspend the virtual function on the source and copy a minimal state (e.g., CPU execution state, registers values, cache memory) to the receiving node so that the virtual function can continue its execution there. Only after that, they copy the rest of the required data. There exist three flavors of post-copy migration, which vary on the way they perform this second step.

The first variant is known as post-copy migration with demand paging method. Once the resumed virtual function tries to access a memory page that is not available on the receiving node, it generates a *"page fault"* and demands that page from the source node. Upon such request, the source node provides the service with the faulted page.

The second method is called post-copy migration with active pushing. According to this method, the virtual function can generate page faults for forcing the source node to transfer faulted pages. However, the overall number of page faults is reduced, as the source node sends concurrently the memory pages to the receiving node even

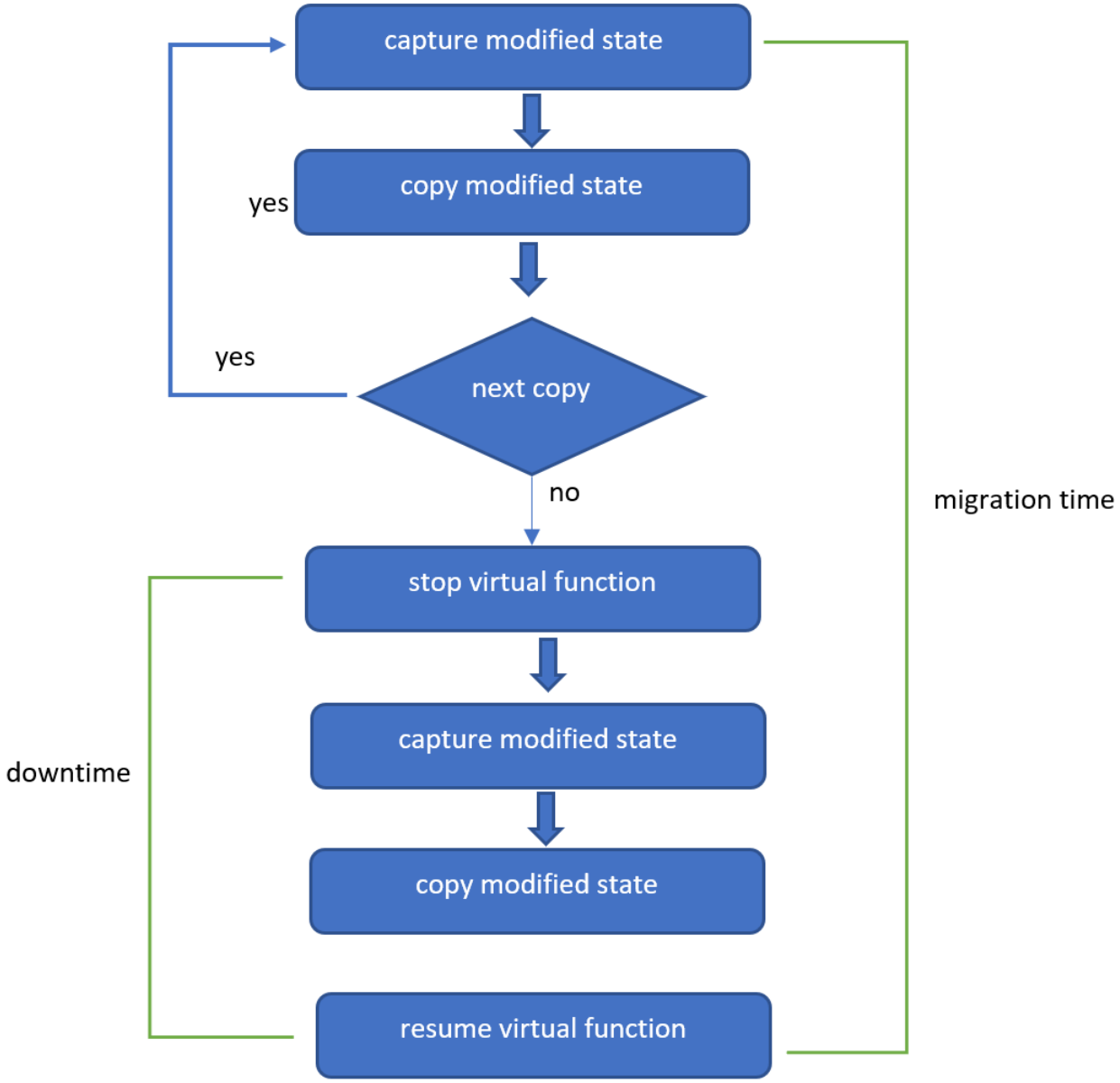

Figure 5·3: Live pre-copy migration model.

if the resumed service has not tried to access them yet. Finally, post-copy migration with pre-paging (Figure 5·4), can further reduce the number of page faults, as the source node actively transmits memory pages that are "close" to the latest faulted page, increasing the probability of transmitting a page that would be requested later on.

5.6 Virtual function migration model

In this section, the migration model for the proposed Virtual Function Chaining model is presented. More specifically, the proposed migration model comprises two modes:

- Cold migration mode (post-active), which is invoked by the orchestrator when an edge device unexpectedly fails, and thus disconnects from the virtual function chains which participated. The reasons for an edge device to fail can vary, from battery drain to computational overloading.
- Live migration mode (pre-active), which is invoked when the overload detection model (*QoS monitoring model*) (described in chapter 3) predicts that an edge node will be overloaded in the next time period.

The two modes are implemented through algorithms 3 and 4, as presented below.

The two algorithms run in parallel and perform the migrations, when required. For the live migrations , the pre-copy mode has been selected. The reason for this decision is that the persistent data of the virtual functions are provisional, especially for streaming services like surveillance analytics. Thus, it does not make much sense to fully transfer the dirty memory pages which store the latest processing frame. Thus, loaded execution programs and business logic can be copied a-priory.

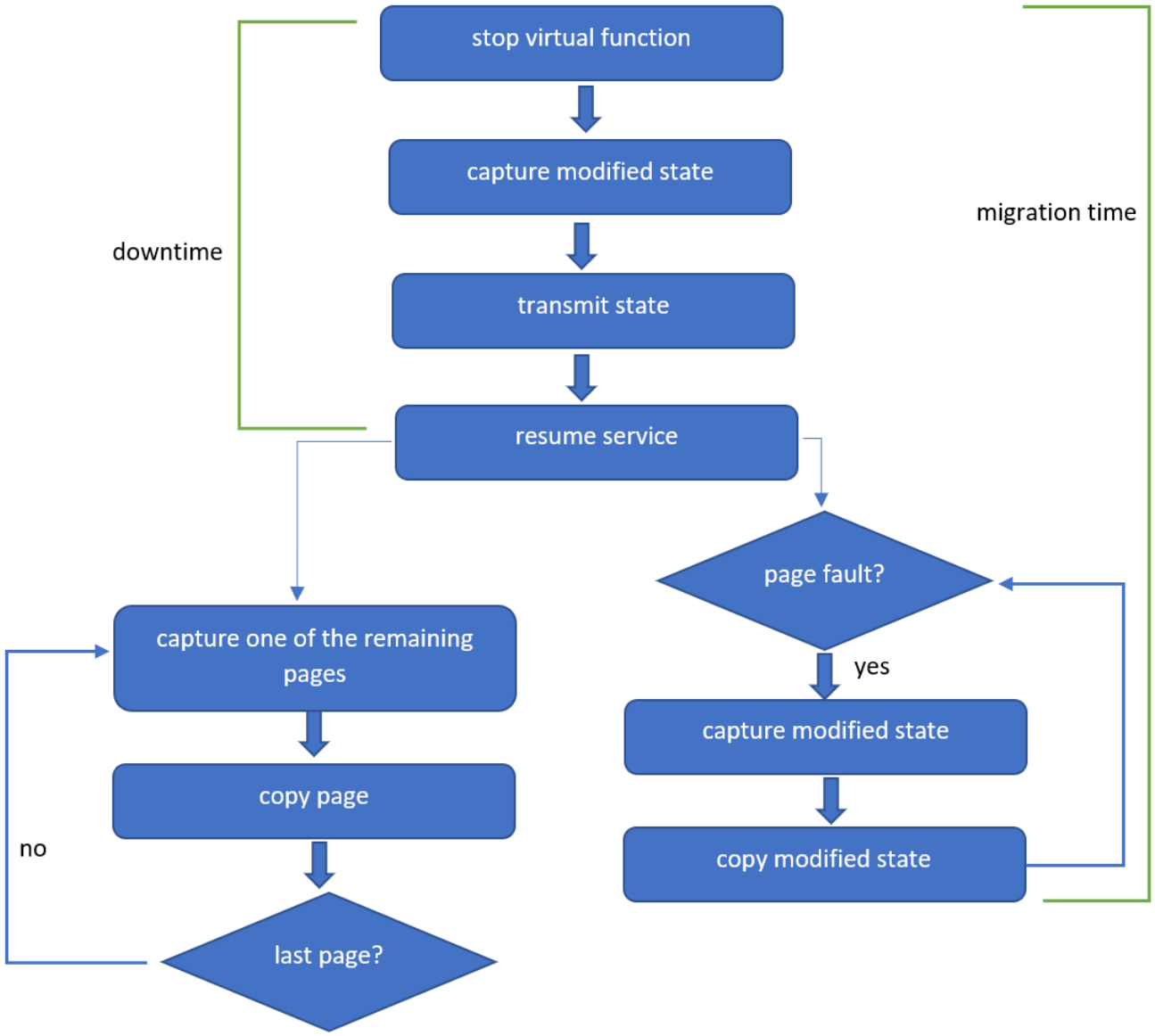

Figure 5·4: Live post-copy migration with pre-paging.

Algorithm 3: Cold migration model.

Input: failed VF
for *network node* n_i **do**
  collect node state s_i
end
1. n_r = *placementAlgorithm*(s_i)
2. *pack a container with the appropriate VF*
3. *orchestrator send the container to the selected node*
4. n_r *unpacks the container and initialize the VF*
5. *orchestrator inform previous and next VF in the chain about the network update*

Algorithm 4: Live migration model.

Input: K, fail probabilities (overload prediction model)
for *network node* n_i **do**
  collect node state s_i
end
for *nodes with failing probability* $\geq K$ **do**
  1. n_r = *placementAlgorithm*(s_i)
  2. *failing node*(n_s) *packs a container with the appropriate VF*
  3. n_s *sends the container to the* n_r
  4. n_r *uppacks the container and initialize the VF*
  5. *orchestrator inform the previous and the next VF in the chain about the network update*
end

Table 5.1: Edge environment model's parameters.

Parameter	Description	Value
g	number of nodes comprising the edge environment	$[100, 200, 300, 400, 500]$
W	network links capacity	$\mathcal{N}(98, 2.32)Mbps$
$c_k(l)$	cost function of a node k for l CPU instructions	$\frac{l^2+\mathcal{N}(2.1,0.5)}{1000}$
m	CPU instructions/sec the node can execute	$\mathcal{N}(10^5, 10^3)$ instructions per sec

5.7 Migration model evaluation

For assessing the performance of the migration model, a set of studies have been performed, both in the simulation environment detained in Chapter 4 and on an experimental setup. The aim of these studies is to:

- Assess the network overhead of migrations,
- Assess the influence of the migration model on the QoS of the deployed services and
- Assess the performance and the overhead of the pre-active mode against the usage of the post-active mode of migrations

5.7.1 Simulation environment

In order to acquire the necessary results, a set of simulation scenarios have been established. While each scenario has certain characteristics, edge environment and $VFCs$ have been modeled under the same principals. More specifically, in Table 5.1, the modeling features for the edge environment are presented, while in Table 5.2, the relative information for the VFC is presented. It is important to mention that the considered environment is heterogeneous.

Table 5.2: VFC model's parameters.

Parameter	Description	Value
n	number of VFs comprise the VFC	$\lfloor \mathcal{N}(8,4 \rfloor)$
cpu_{load}	instructions per frame for a specific VF	$\mathcal{N}(10^4, 10^3)$ instructions/frame
$output$	VF output size per frame in bytes	$\mathcal{N}(10^5, 10^4)$ bytes

5.7.2 Simulation results

A. *Dedicated* edge environment

The first set of simulation scenarios were established by modeling the edge nodes *'behavior'* with constant probabilities. More specifically, an environment with g nodes has been simulated. *Dedicated* refers to the fact that the simulated edge environment does not host any other tasks or jobs, aside from the deployed VFs. For simulating the overload of a node, the following approach has been applied. Each node has been supplied with a probability $P = P_{leave} + P_{overload}$, where P_{leave} expresses the probability the node suddenly fails and $P_{overload}$ expresses the probability a node exceeds $K\%$ of its computational capacity, where K is a constant. It is important to mention that exceeding the K of the computational power does not mean that a live migration will be triggered. Triggering of a live-migration is based on the accuracy of the overload prediction model (*QoS assessment*), as presented in Chapter 3.

Regarding the services' demand, as in paragraph 3.3.2, three different modes have been considered. Namely, low ($\lambda = 54$ requests/hour), normal ($\lambda = 83$ requests/hour) and high ($\lambda = 118$ requests/hour) demand modes have been simulated. More specifically, the arrival time for a service request has been modeled as a Poisson process, with different λ for each mode.

Two main scenarios have been considered (Table 5.3). According to the first scenario, the overload prediction model is not deployed. Thus, a (cold) migration is

Table 5.3: Deterministic simulations characteristics.

Simulation scenario	Description
1	Overload prediction model not deployed. Only cold migrations are considered.
2	Overload prediction model is deployed. Both cold and live migrations are considered.

only triggered when a node, hosting a virtual function, fails unexpectedly, either due to overload or due to disconnection from the network. The second scenario deploys the overload prediction model (when the node's load exceeds K) and in parallel enables the live migration model, as previously discussed.

The results of the simulations are presented in the following figures. More specifically, Fig. 5·5 to Fig. 5·9 presents the services downtime (as percentage % to the overall simulation time) for the different number of edge nodes, when the users' demand rate is set to normal ($\lambda = 83$ requests/hour).

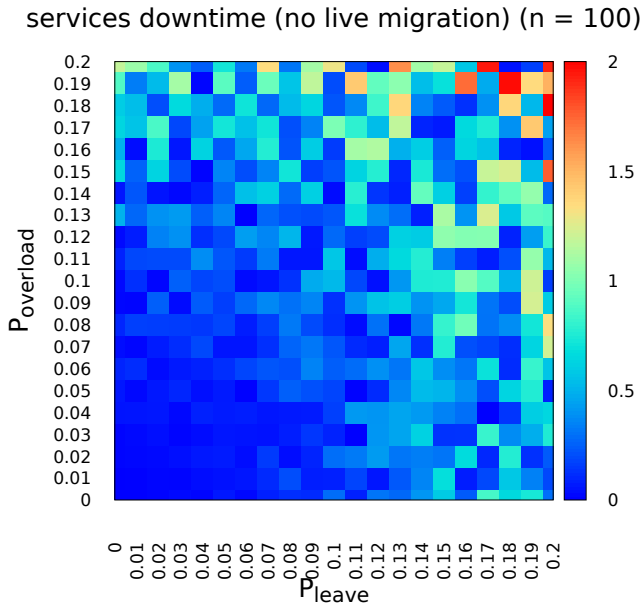

Figure 5·5: Services' downtime (%) without the deployment of overload prediction model (n=100).

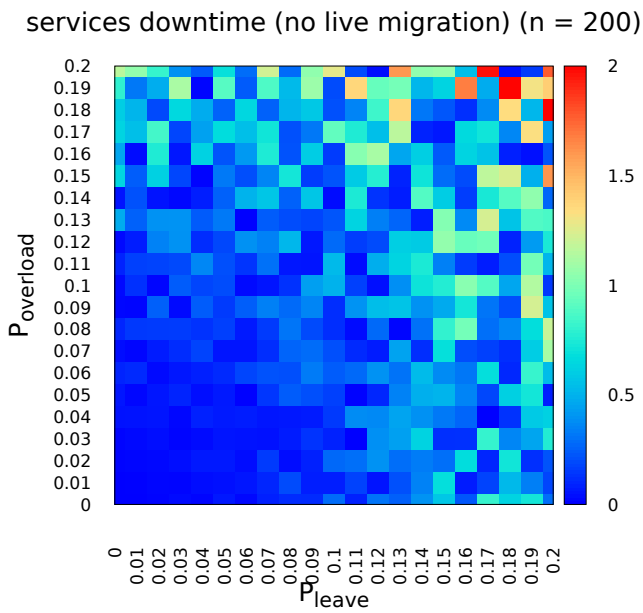

Figure 5·6: Services' downtime (%) without the deployment of overload prediction model (n=200).

The next scenario applies the *QoS assessment model*, which triggers a live migration whenever the load of a node exceeds K. Fig. 5·10 to 5·14 presents the influence of P_{leave} and $P_{overload}$ for different number of edge nodes. Comparing the presented heatmaps, the influence of the *overload prediction model* is obvious. On average, services' downtime is improved by 34.2%.

Similarly, the total data volume transmitted over the network for supporting the migrations has been considered. Figure 5·15 presents the total transferred volume when overload prediction model is not deployed ($n = 100$, users' demand rate set to normal), and Figure 5·16 the relative one with the overload prediction model deployed. As expected, the volume of the transmitted data for supporting VF migrations is greater (19.1%) when the overload prediction model is deployed, as more migrations take place. Thus, while the overload prediction model improves the QoS for the deployed services, it produces more network data traffic.

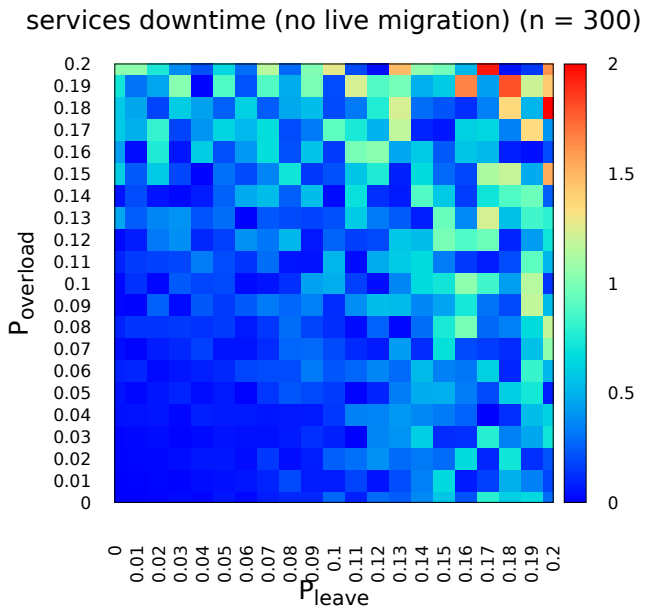

Figure 5·7: Services' downtime (%) without the deployment of overload prediction model (n=300).

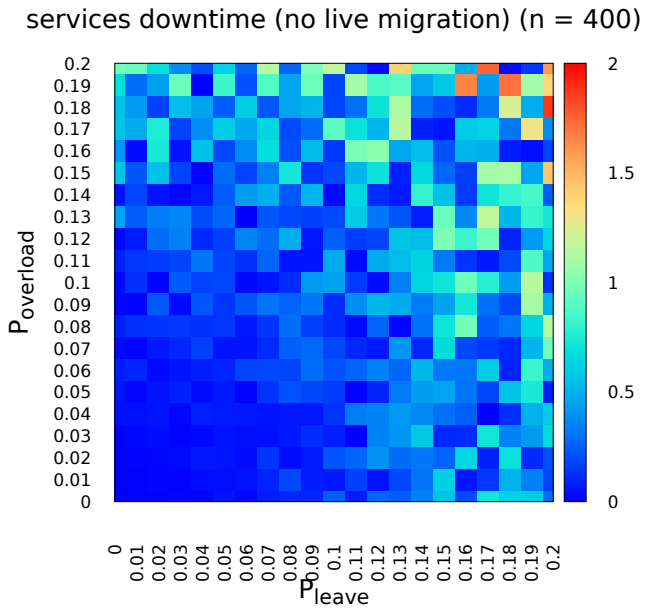

Figure 5·8: Services' downtime (%) without the deployment of overload prediction model (n=400).

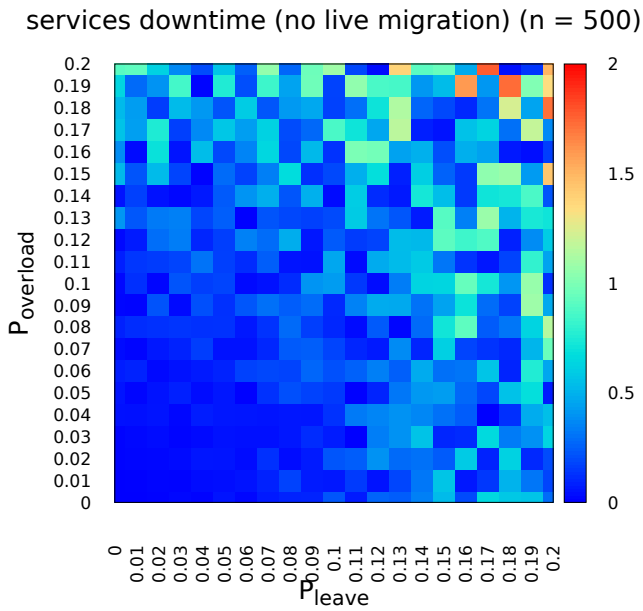

Figure 5·9: Services' downtime (%) without the deployment of overload prediction model (n=500).

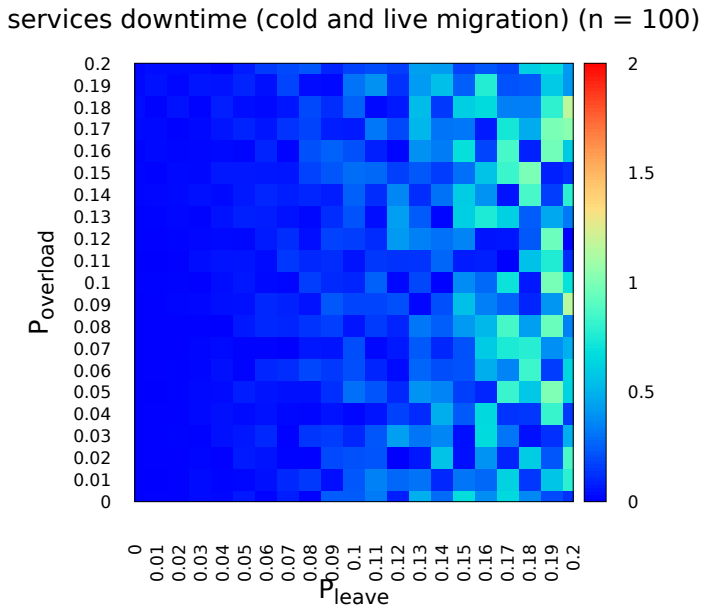

Figure 5·10: Services' downtime (%) with the deployment of overload prediction model.

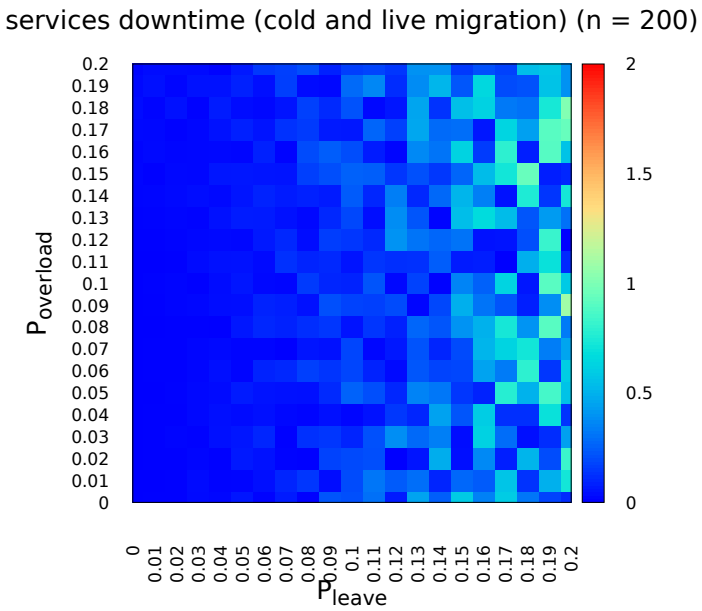

Figure 5·11: Services' downtime (%) with the deployment of overload prediction model.

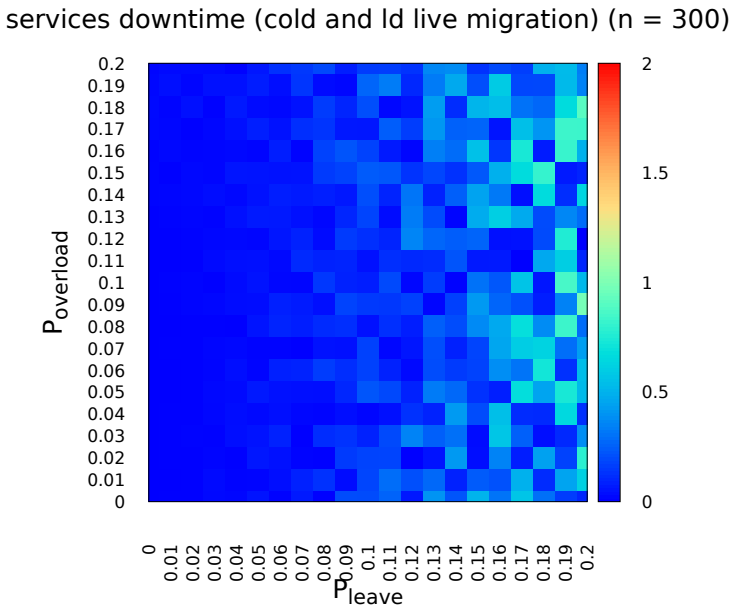

Figure 5·12: Services' downtime (%) with the deployment of overload prediction model.

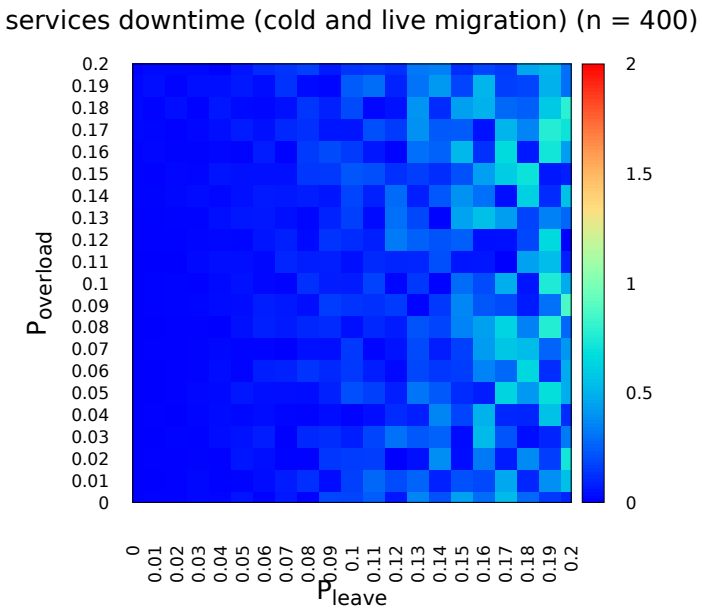

Figure 5·13: Services' downtime (%) with the deployment of overload prediction model.

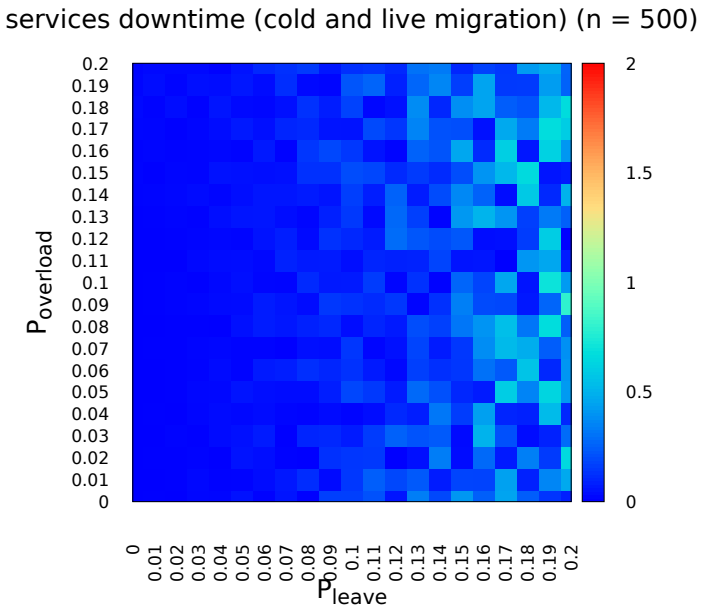

Figure 5·14: Services' downtime (%) with the deployment of overload prediction model.

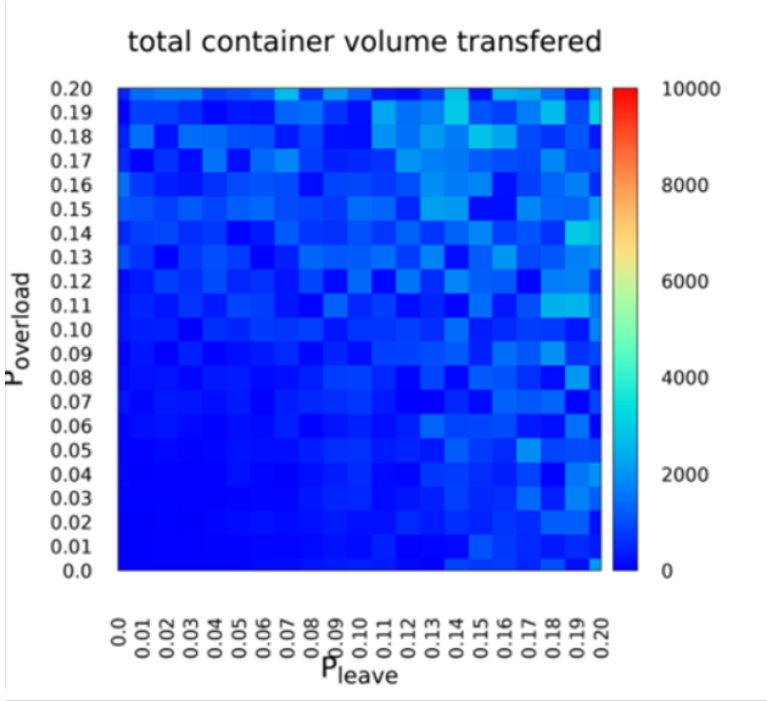

Figure 5·15: Total data (MB) transmitted without the deployment of overload prediction model.

Fig. 5·17 and Fig. 5·18 presents the average downtime of the deployed surveillance services along with the absolute number of migrations respectively, for different number of edge nodes, when $P_{leave} = P_{overload} = 0.1$. From these figures, one can conclude that by increasing the number of the available nodes, downtime is improved and the required migrations decrease, as the placement algorithm can, statistically, detect more *stable* nodes to migrate a VF, when required. In addition, increasing the users' demand rate increases the average downtime, as well as the number of required migrations. More important, the deployment of overload assessment module improves downtime while increasing the required migrations.

B. *Generic* edge environment

For the next set of simulations, an attempt has been made to simulated the workload of a node more realistically. For this, each node $\mathcal{D}$ in the edge environment is supplied

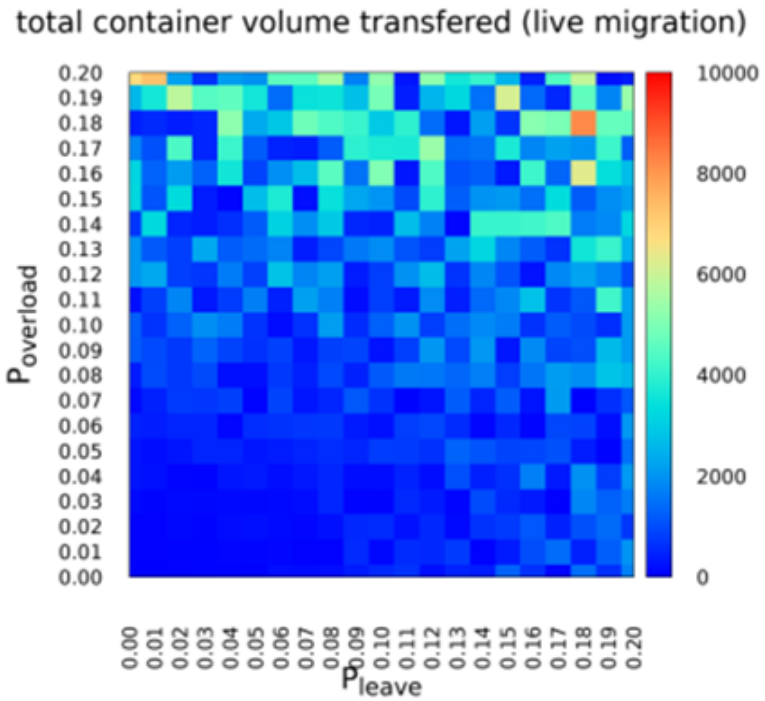

Figure 5·16: Total data (MB) transmitted with the deployment of overload prediction model.

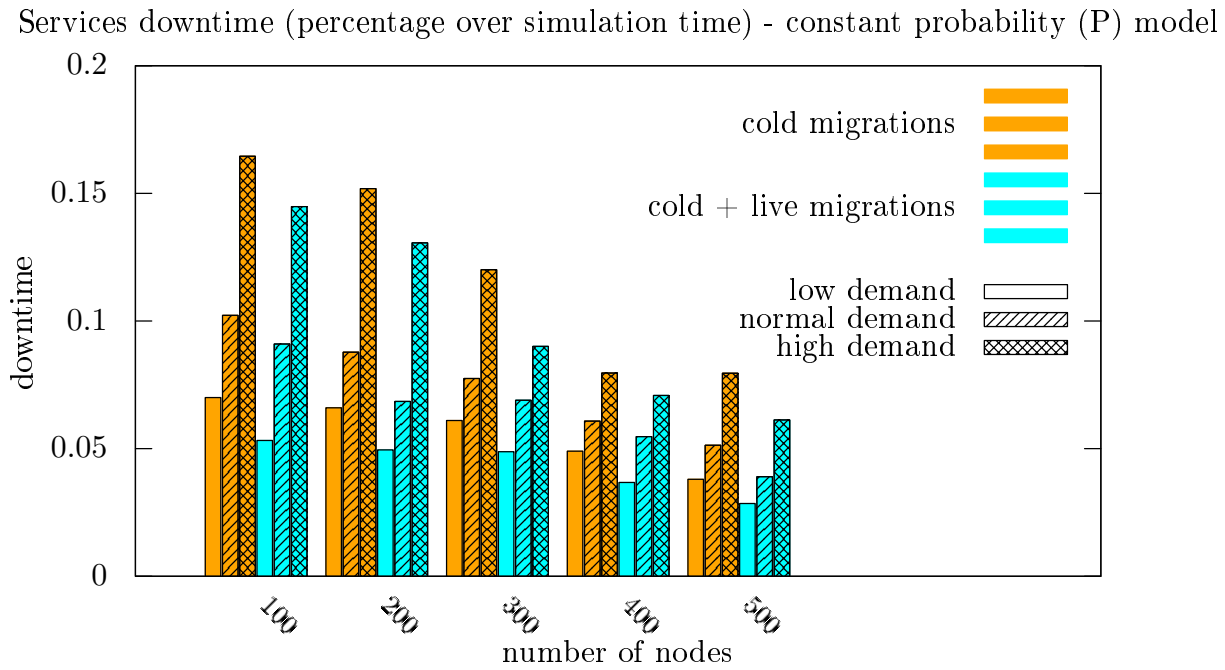

Figure 5·17: Services' downtime (%).

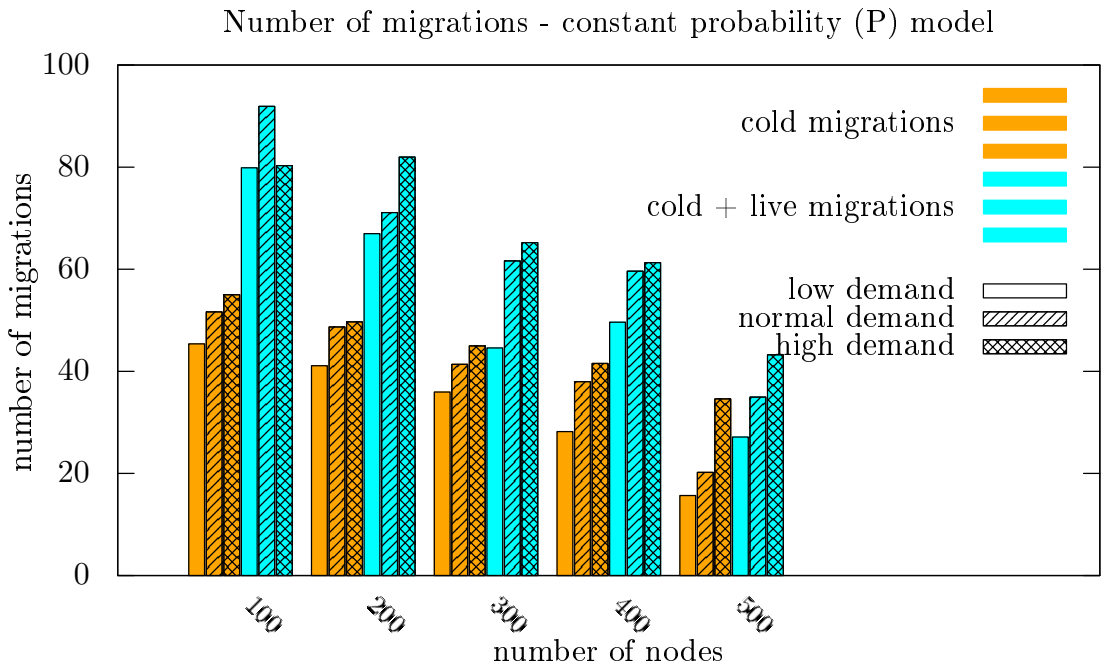

Figure 5·18: Number of required migrations.

with a probability density function $\mathcal{H}(t)$, which describes the probability a job $\mathcal{J}$ arrives in the node at time t. Each job $\mathcal{J}$ is characterized by a CPU load, a memory RAM load and its duration t. If a job $\mathcal{J}$ is undertaken by a node, it consumes its computational resources for t secs. If not, it enters a FIFO queue, until the necessary resources are free. A VF has the same priority as the other jobs, with the difference that it does not wait in the queue, and the orchestrator seeks for another node candidate.

This model, which mimics the *'behavior'* of an edge node more realistically, allows for the deployment of the *QoS assessment model*, as described in the previous chapters (based on the trained LSTM models). Thus, $P_{overload}$ is now simulated by a stochastic process. More precisely, the arrival time of a new task (different than VFs) is modeled as a Poisson process, with probability mass function $P(k \quad jobs \quad in \quad simulation \quad time) = e^{-\lambda}\frac{\lambda^k}{k!}$, where k, λ are randomly selected for each different edge node during the simulation setup and equals to $k = \lfloor \mathcal{N}(8,5) \rfloor$ and $\lambda = \mathcal{N}(5,2)$. When a node undertakes a new task, the processing time of a VF increases. When this time exceeds a certain threshold, implied by the required processed frames per second, the *overload* event is triggered, along with the relative migration. For the same set of simulations, $P_{leave} = \mathcal{N}(0.1, 0.005)$. This probability density function is probed 10 secs after the deployment of a VF on an edge node (either initial installation or installation due to migration).

Similarly with the previous section, Fig. 5·19 and Fig. 5·20 presents the average downtime of the deployed surveillance services along with the absolute number of migrations respectively, for different number of edge nodes.

While the absolute values differ from the *'dedicated'* edge environment scenario, the trend of the results remain the same, for all three user demand rates.

The next step is to assess the optimum probability K, which acts as a threshold

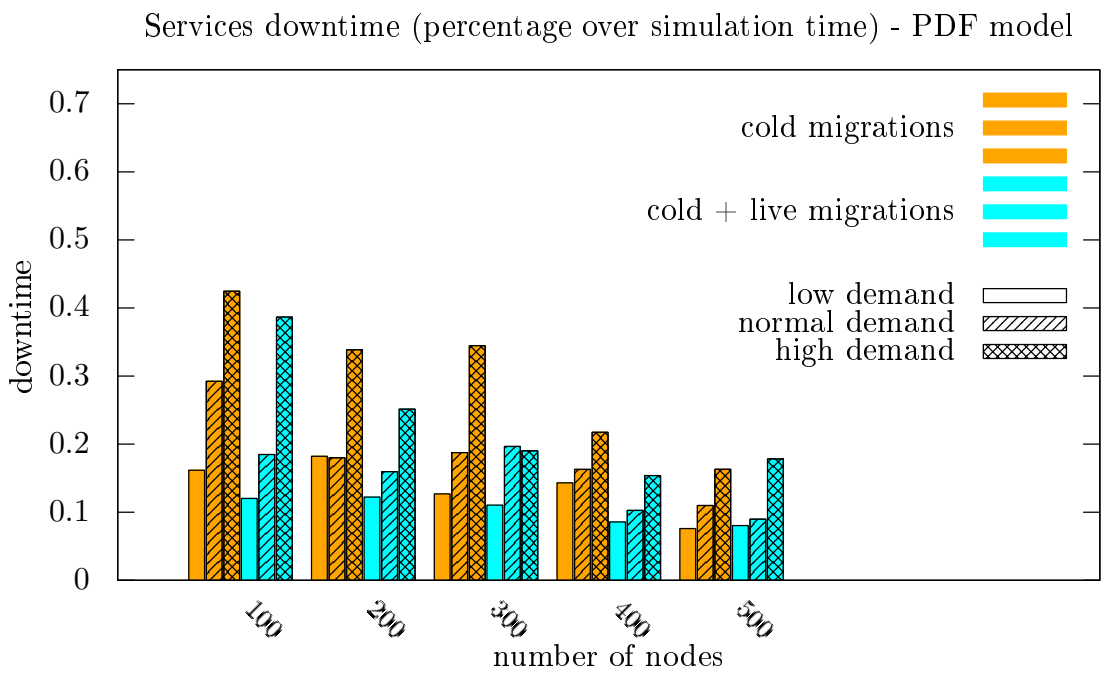

Figure 5·19: Services' downtime (%).

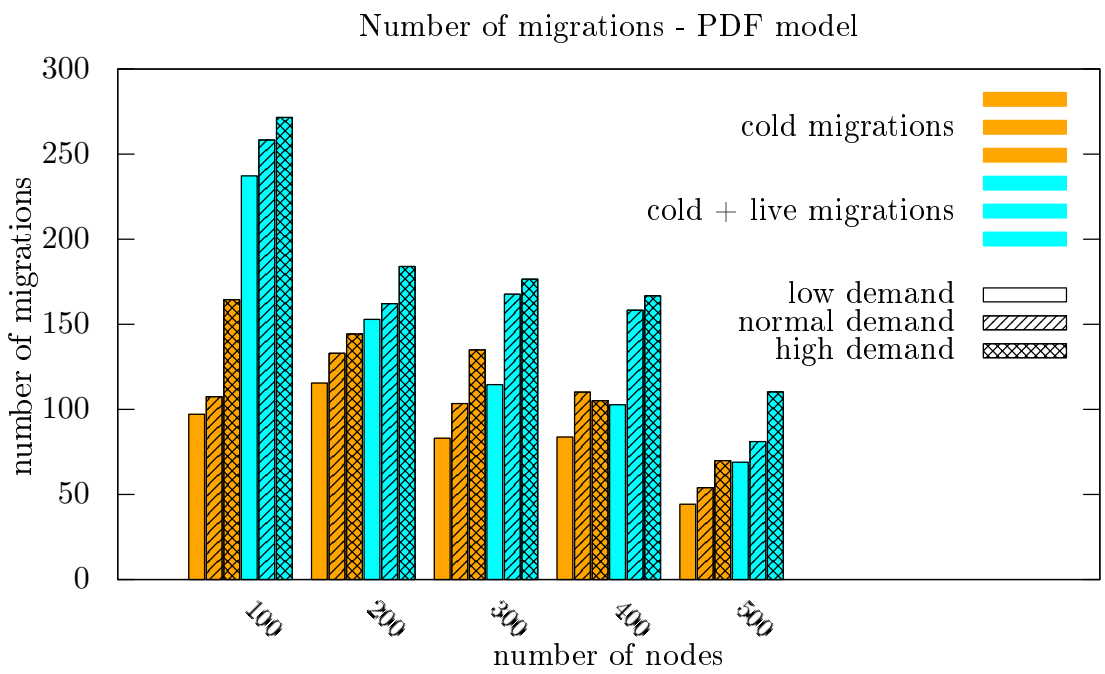

Figure 5·20: Number of required migrations.

for triggering a live migration, based on the output of the failure assessment model. Figure 5·21 and Figure 5·22 present the results on a set of simulations performed for this scope. For these simulations, $P_{leave} = 0.1$, while $P_{overload}$ is modeled with the previously described stochastic process and users' demand was set to normal. Finally, different sizes of the edge environment have been considered.

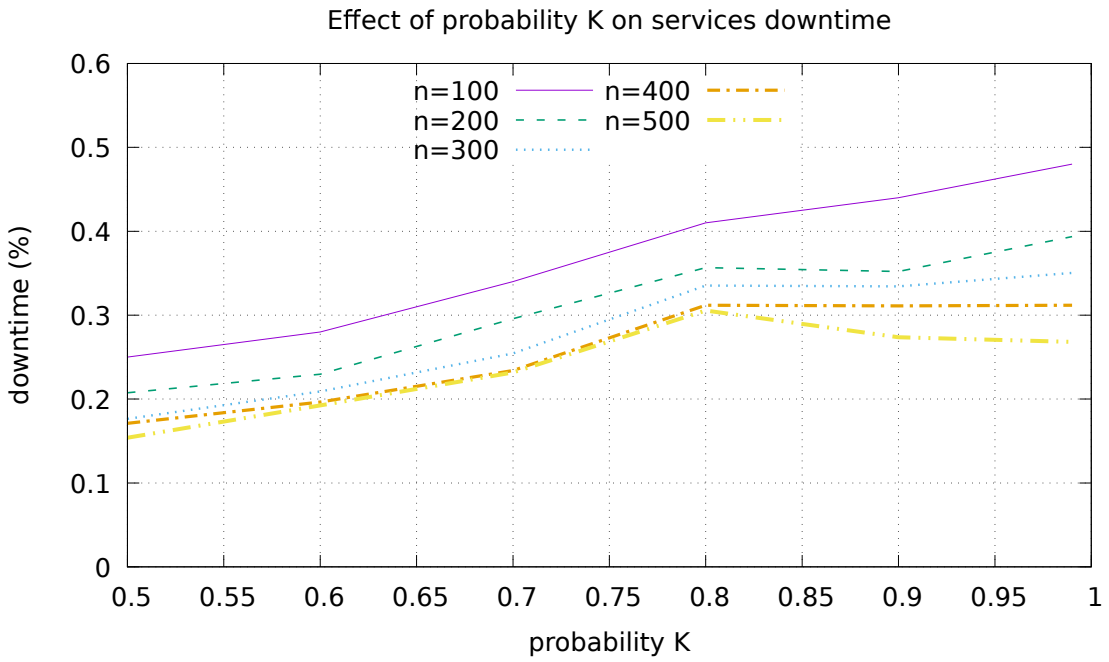

Figure 5·21: Services' average downtime for different values of K and n.

As one can notice, smaller values of K cause more migrations and has a benefit effect on services' downtime while higher values of K increases downtime and decreases generated volume traffic.

For the live migrations, it is interesting to assess the timing of each different phase of the process. The results for this task are presented in Figure 5·23. The data transmission phase is the most timely one, especially due to the WAN connection links, which were simulated in the environment.

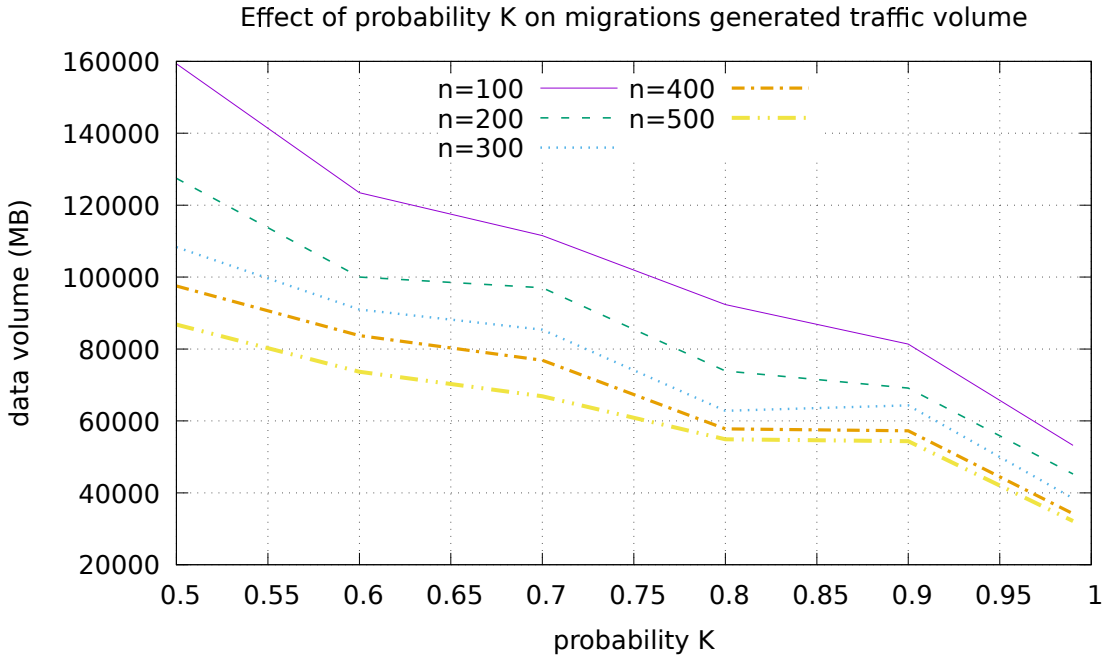

Figure 5·22: Volume produced by migrations for different values of K and n.

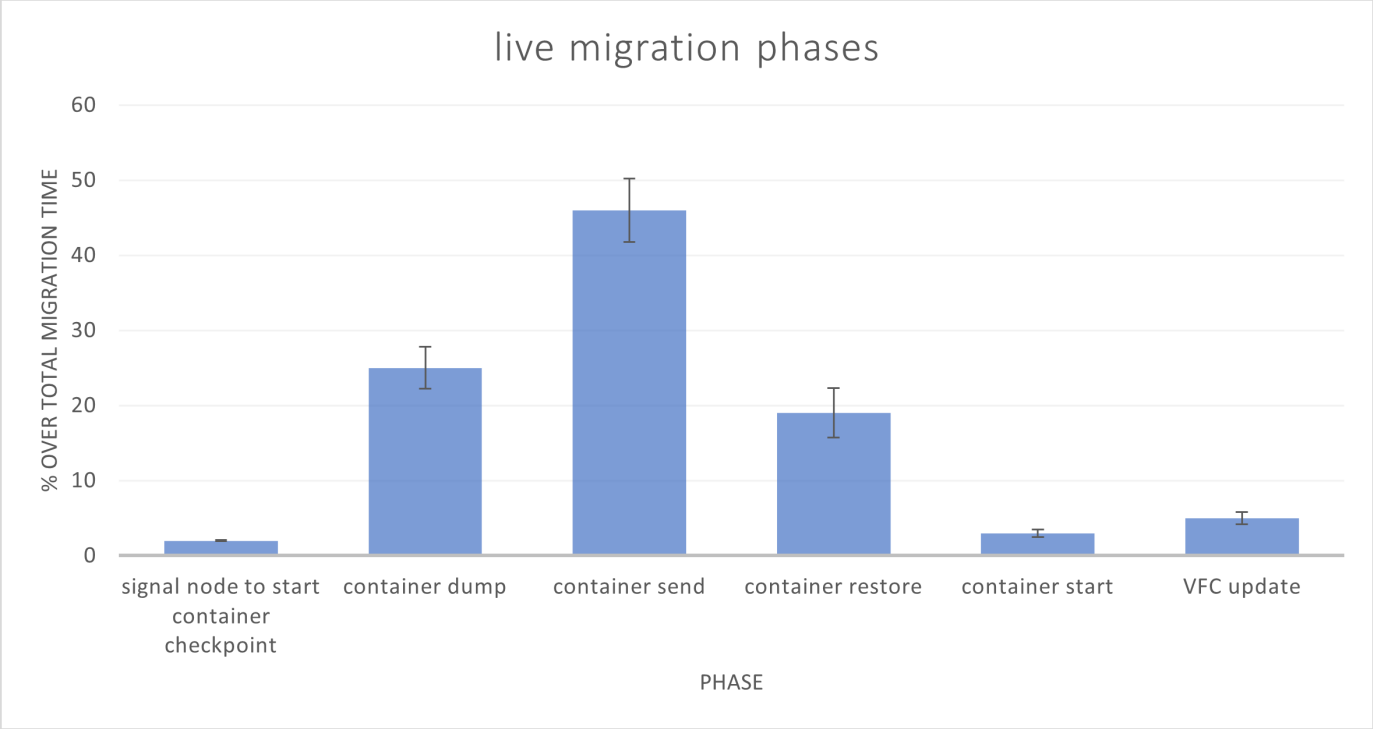

Figure 5·23: Timing of the live migration phases.

5.7.3 Comparison with similar frameworks

On top of the aforementioned studies regarding the scalability of the proposed VFC migration model, the required time for deploying a service on an edge network with n devices is analyzed. Without loss of generality, lets assume an edge network with wireless devices, based on WiFi 802.11g connections (S Mbps actual bandwidth). Additionally, let j be the number of VFs to be deployed using c GB containers each.

The deployment of a service following the VFC model comprises four steps:

- **Initial probe of edge devices available resources.** If a kb is the size of the probing message, then it would require approximately $\frac{a \times 8}{S \times 10^3}$ secs for the j devices to transmit the data.

- **Placement problem solving.** Based on (Hedengren et al., 2012), APOPT solver requires polynomial time to solve a mixed-integer problem with one non-linear equation.

- VFs **deployment.** For each one of the chosen edge devices, it would require $f \times \frac{c \times 8 \times 10^3}{S}$, where f is a coefficient which expresses the delay which will be caused by reaching the limit of the output bandwidth of the VFO node.

- **Monitoring phase.** Every 30 secs, the j selected edge devices, hosting a VF each, are informing the VFO about their available resources, enabling the QoS monitoring service to function. This phase requires $j \times a$ kb of data to be transferred in the network every 30 secs, with each transmission to require $\frac{a \times 8}{S \times 10^3}$ secs.

Based on the described network times, as well as the times acquired for solving the placement problem (using an Intel i7 2.8GHz (8core) on 8GB of RAM), the results presented in Fig. 5·24 has been obtained. The values of the variables were: $j = n/2$, $c = 1.2GB$, $a = 2kb$, $S = 100Mbps$.

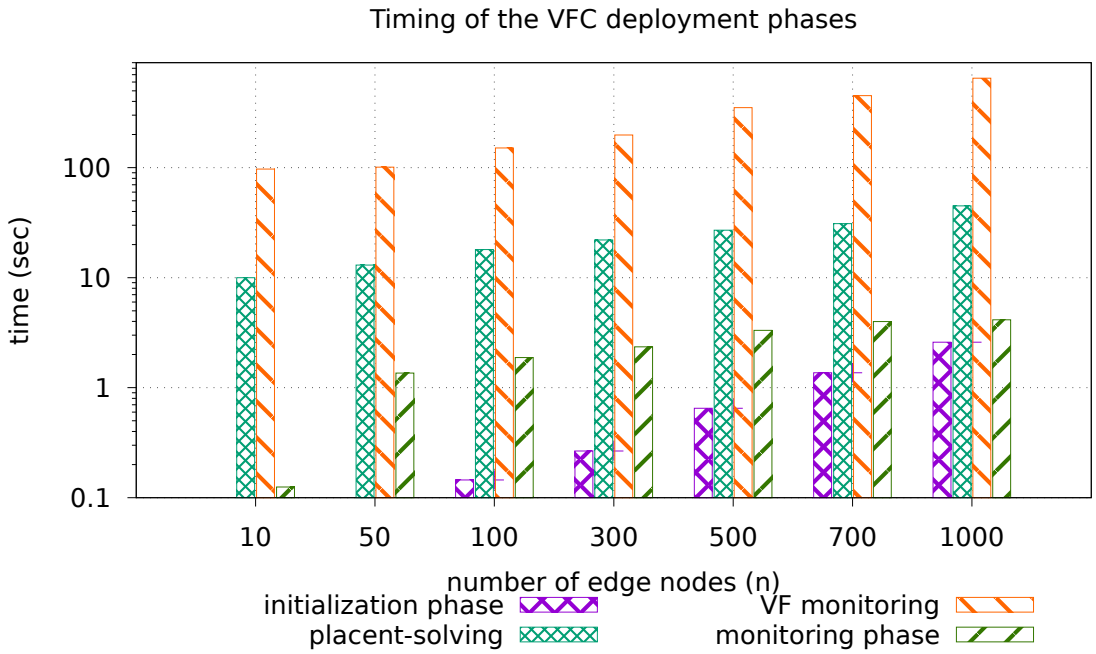

Figure 5·24: Timing of the different VFC phases.

One can notice that even when using 1000 edge devices and 500VFs, in order to deploy a service, the required time to select the optimal nodes is kept relatively small, enabling the efficient scaling up of the model.

Kubernetes framework

Kubernetes (Burns et al., 2022) is an open-source container orchestration tool, which quickly after its introduction, became the de facto standard for managing large container deployments. Kubernetes support by default orchestration and autoscaling of containerized services, based on the users' demand. Aiming to evaluate the performance of the autoscaling capacity of the proposed VFC model, a Kubernetes cluster has been build using the Raspberry Pi cluster as worker nodes, based on the blueprint proposed in (Kayal, 2020).

The testbed comprise a PC (Intel x64 architecture) serving as Kubernetes master node and the 8 Raspberry Pi devices described in Section 3.3.2. *Service A* has been

deployed on the cluster and a user simulated the demand alterations by changing the requested fps processed. Total edge environment cost and number of deployed VFs have been considered when comparing the two approaches.

VFC Autoscaling capacity evaluation

A set of experiments for assessing the VFC model's capacity to autoscale the deployment of the VFs depending on the users' QoS demand has been conducted. According to the experimental scenario, *Service A* has been deployed on both VFC and Kubernetes frameworks. The simulation run for 60 minutes on each framework. Within the simulation time, the required QoS (requested fps processed) was randomly changed every 5 minutes, within the range $[5, 20]$. During the first 5 minutes, the requested fps was set to 0, aiming to assess the zero-demand footprint of the solutions under comparison. Total edge environment cost, as well as total number of deployed containerized VFs have been probed and the relative results are presented in Figures 5·25 and 5·26.

From the reported results we can observe that the VFC model can follow the demand changes more efficiently compared with Kubernetes, even if the improvement is small. As far as the total edge network cost is concerned, VFC presented a reduction of 12.54% compared to Kubernetes, averaging the cost over the 60 minutes experiment.

Results on experimental setup

After exploring the simulator for assessing the influence of live migrations on the services status, the migration models have been implemented in the experimental setup, consisting of six Raspberry Pi 3 and two Raspberry Pi 4. The service deployed was the same as the one described in chapter 3, and run for 6000 seconds.

While the improvement of the average value of the processed fps is rather small

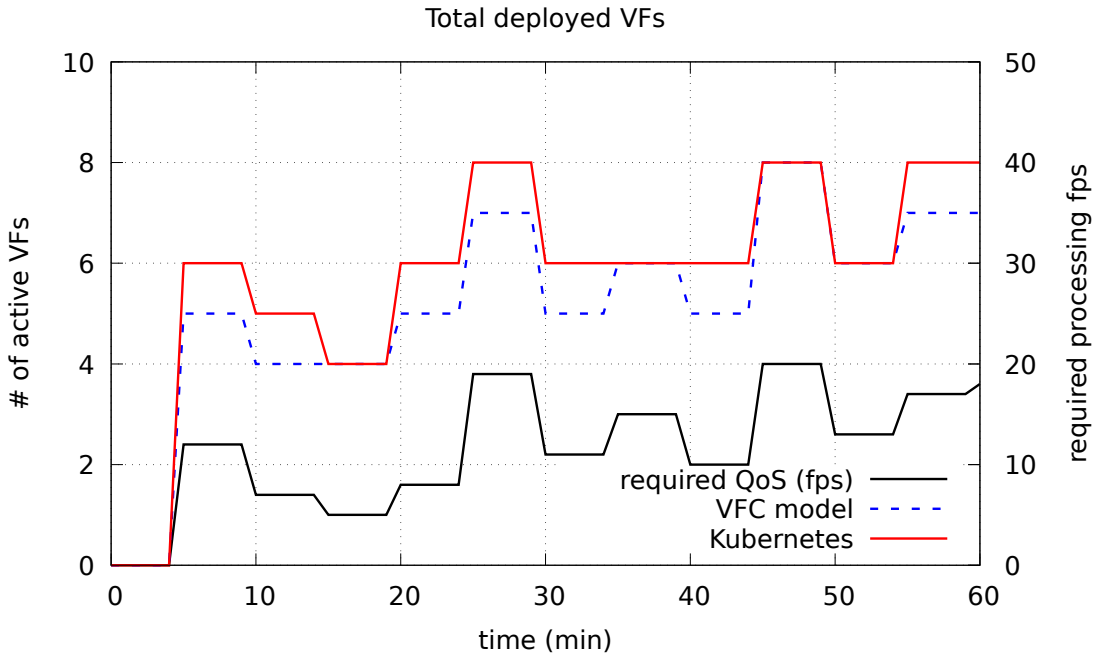

Figure 5·25: VFC model vs. Kubernetes - deployed VFs.

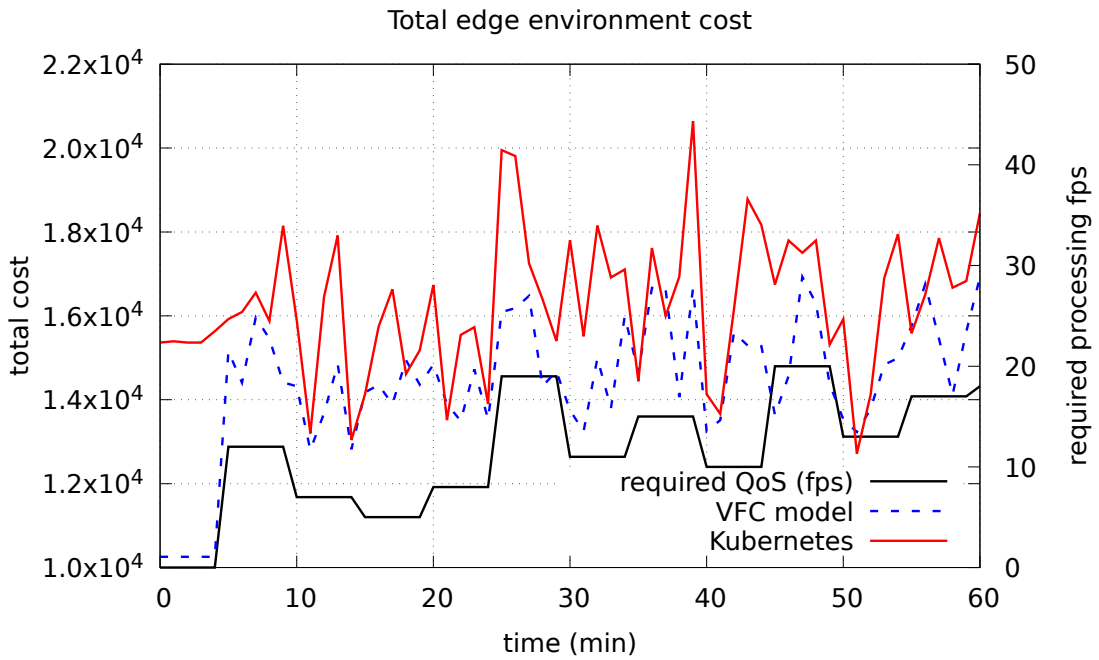

Figure 5·26: VFC vs. Kubernetes - Total environment cost.

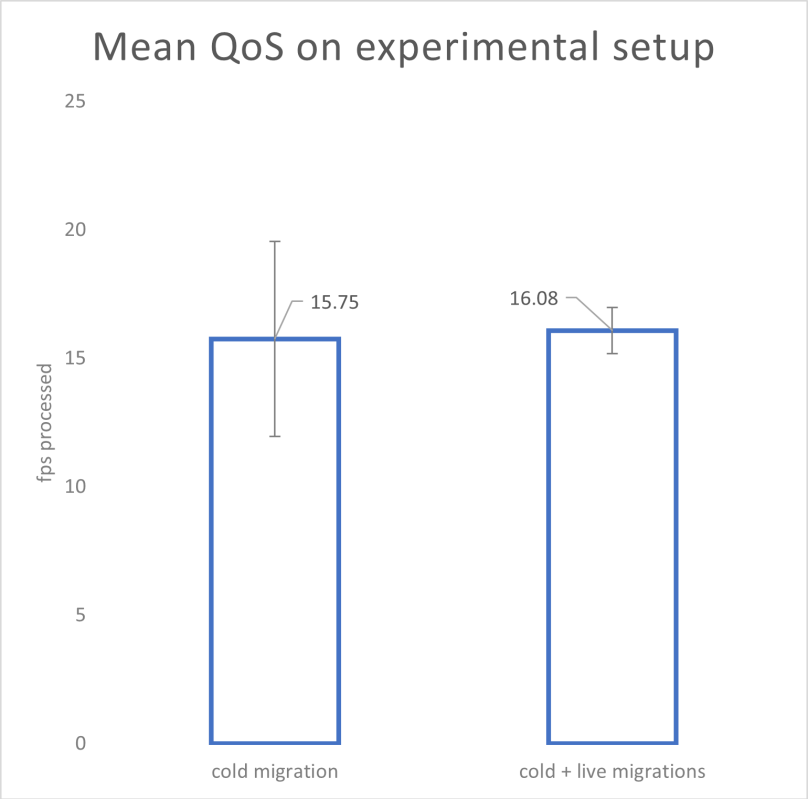

Figure 5·27: Experimental results on QoS.

(2%), when using the live migrations model, the fluctuation on the QoS is significantly decreased (76.3%), making the service much more stable and robust (Figure 5·27).

5.8 Conclusions and Discussion

Within this chapter, the migration model for the VFC framework has been presented. After presenting relevant technologies and methodologies for edge-based migration services, the designed approach for the VFC migration model is detailed.

Two types of migrations have been considered, cold migrations and live migrations. In detail, cold migrations refer to the scenario according to which an edge node hosting a VF fails unexpectedly, while live migrations refer to the scenario according to which an edge node transfers its deployed VF to another edge node, due to possible node failure.

The algorithm for supporting cold migrations is based on a monitoring mechanism by the VFO edge node. According to this mechanism, VFO probes the binary status of all VFs (*up and running / not responding*). As soon as a non-responding VF is detected, VFO executes the placement algorithm for calculating the most appropriate node to undertake the failed VF. The next step includes the deployment of the VF to the selected node and the update of the VFC according to the new establishment.

As far as live migrations are concerned, the migration model works in parallel with the VFC QoS monitoring model. According to this pipeline, VFO probes the status of the edge nodes utilized in the VFC and assess their probability to fail in the next period. As soon as the model detects a failing node, it initiates the live migration model. The next step involves the utilization of the placement algorithm for detecting the most appropriate node for undertaking the VF from the failing node and of cource the actual migration steps from one node to the other.

The VF migration model plays a important role on the robustness of the overall

framework, as it imposes a self-healing mechanism against edge nodes failure. In contrast to cloud infrastructures, edge environments appear large fluctuations to the capacity and availability of the processing nodes. In order to meet the different characteristics of such environments, we have proposed a migration model with light demands, in terms of computational requirements, which improves the overall QoS of the deployed services, without requiring an exceed amount of computations.

Finally, comparing the capacity of the VFC model to auto-scale and self-heal, the experiments described in this chapter against similar distribution frameworks (Kubernetes) showed that the proposed model has the capacity to operate efficiently.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Edge computing is expected to be an important part of the AI industry during the next few years. Its advantages lie not only on the proximity with the processing data, but also on the data protection, privacy and safety issues, which are debatable on the Cloud Computing paradigm. This work proposes a novel concept for enabling real-time AI applications on an Edge network, such as video analytics. Our proposal is based on the $VFCs$ which are used to distribute an AI application across the edge network on an scalable fashion. After providing a mathematical model for the proposed system, the results of a real-case scenario are reported, where the system has been implemented and tested in various conditions. A caching mechanism is also described, which extents even further the capacity of the system. Finally, a migration model suitable for VF management is designed, implemented and evaluated, both in simulation environment and in a real testbed system. The experiments have provided evidence that the proposed model can be used to undertake heavy-load AI applications and handle them in real-time, under QoS constraints.

The proposed model has been applied on a video analytic service, which belong in the family of streaming applications, in terms of data generation. The nature of the streaming applications matches the characteristics of the proposed VFC model, which partially explains the really good performance of the model against other distribution approaches. Thus, while it is anticipated that the proposed model would

be outperformed by generalized distribution schemes (i.e., Apache Spark ©) on non-streaming applications, streaming applications from other domains, like sentiment analysis on data coming from social media are expected to have similar performance benefits as the tested surveillance service. Deploying such applications on the VFC model is part of our future plans.

For supporting the evaluation of VFC model, a simulation environment has been designed, implemented and deployed. More specifically, $VFCSIM$ can be used to simulate both virtual machines and containers, as virtualization and dockerization technologies are recently considered not only to the cloud but also to the edge computing paradigm, an abstract approach for developing services is required. VFC model can support this design approach, as it is a simulation framework which can support modellers, engineers and researchers to test architectures and solutions based on VFs.

Fluctuations and diverse characteristics of edge/fog environments dictates the deployment of self-healing services, in order to support processing devices' registering and/or un-registering from the environment. As migration is considered one of the most appropriate solutions for handling edge devices disconnections from the edge network, a migration model for VF has been designed, implemented and deployed on-top of the proposed VFC model. This migration model considers both ***cold*** migrations, for supporting sudden disconnections of edge devices, and ***live*** migrations for offloading edge devices with high utilization rates. By using $VFCSIM$, the proposed migration model has been evaluated under different scenarios. The relevant results showed that the migration model can improve the QoS characteristics of a deployed service. The same conclusions can be drawn by the experiments conducted on a real testbed with eight edge devices (Raspberry Pi devices). More specifically, according to the results from these experiments, the variation of the processed frames

per second of a VSS is significantly reduced.

6.2 Future Work

Within the lifetime of this PhD project, a novel scheme for distributing AI tasks on a heterogeneous edge network has been designed, implemented and evaluated. For supporting the proposed scheme, the first step was to build a simulation environment ($VFCSIM$) tailor-made for VFC simulations. By using this simulation environment, VFC supporting models, like caching and migration models have been tested.

The next step was to implement on a real test-bed the proposed model and compare it against nowadays common practice (cloud services) and generalized distribution schemes (Apache Spark ©).

About the relevant future work, it will be focused on four main directions. First, the deployment of the proposed model on a large-scale real environment will allow the comparison of the scale-up characteristics of the proposed model with the simulation results. In addition, scaling-out the proposed model with other streaming applications, aside surveillance services, will demonstrate the general character of the proposed model. For example, real-time processing of data generated from social networks (e.g., sentiment analysis on Twitter data) is anticipated to be perform relatively good.

Second, throughout the analysis of the model presented in the previous chapters, all of the supporting services of the proposed VFC model (VF placement, migration, edge nodes monitoring, etc.) are hosted on the network's orchestrator node. Our plan is to produce a fully distributed version of the VFC model, with the capacity to deploy these services decentralized, by utilizing graph algorithms like leader election and distributed spanning trees detection.

Third, a communication protocol among different virtual function chains will allow

the execution of surveillance services like *person re-identification*, according to which some temporary results on person identification must be shared to other edge nodes, in order to identify whether a person is already included in the system or he/she is appears for the first time.

Finally, about the simulation environment ($VFCSIM$) , an integrated user interface for designing networks, devices and VF is expected to facilitate the work of the modellers. On the same direction, the integration of libraries with well-established devices and communication protocols will result to a more concrete simulation framework.

To conclude, VFC model can act as an AI accelerator at the Edge, enabling the execution of complex and computational heavy tasks.

Appendix A

Optimization Theory

Resource management problems are usually expressed in the form of constrained optimization problems, where a predetermined objective function (objective) must be optimized under specific constraints that dictate its solution. The goal of resource management problems is either to maximize a core efficiency or to minimize a core cost, related to the amount of resources consumed to ensure service quality. However, depending on the solution to the problem (ie, objective function, resources and constraints), resource management problems can be solved with different learning tools. For this reason, this chapter presents the basic optimization tools used in this PhD thesis, with the aim of presenting a comprehensive solution framework for the optimization problems that are commonly encountered. In more detail, the following are presented: The Optimization Theory, the Variational Inequalities Theory, along with some basic concepts from the Game Theory.

A.1 Optimization Theory

First, some principles from the optimization theory are presented (Floudas, 2009). In more detail, the definition of an optimization problem is presented, some principal definitions and optimization conditions, and finally the Euler-Lagrange equation.

A.1.1 Definition of Optimization problem

Unconstrained Optimization problems

Before considering the constrained optimization problem, which is the most common for resource management problems, the corresponding unconstrained optimization problem is presented. More specifically, the general form of such a problem has the following form:

$$\min_{x \in \Re} f(x) \tag{A.1}$$

where the vector $x = (x_1, ..., x_n)$ is the optimization variable of the problem and the function $f : \Re^n \rightarrow \Re$ corresponds to the objective function to be optimized (or the cost function for specific minimization cases). A vector $x*$ is called total Optimal or otherwise the solution of the problem A.1, if it has the lowest objective value of all the vectors in $\Re^n$, that is, if for every $z \in \Re^n$ holds that $f(x*) \leq f(z)$. Additionally, a vector x^+ is called a local optimal if there is $\epsilon > 0$ such that $f(x^+) \leq f(z)$ with $\|x^+ - z\|_2 \leq \epsilon$. It is stated that an optimization problem can be transformed into a minimization problem A.1, if the opposite objective function $-f$ is considered.

Constrained Optimization problems

A constrained optimization problem is defined in its general form as follows:

$$\begin{aligned} & \min_{x \in \Re} f(x) \\ s.t. \quad g_i(x) \leq 0, for \quad & i \in [1, m] \\ h_j(x) = 0, for \quad & j \in [1, n] \end{aligned} \tag{A.2}$$

where the vector $x = (x_1, ..., x_n)$ is the optimization variable of the problem in the set $\Omega \subseteq \Re^n$, the function $f : \Re^n \rightarrow \Re$ corresponds to the objective function to be optimized (or the cost function for specific minimization cases), while the functions $g_i, h_j : \Re^n \rightarrow \Re$ are the constraint functions of inequalities and equalities, respectively.

If the constraints are absent, i.e. $m = p = 0$, problem A.2 is simplified to an optimization problem without constraints for $\Omega = \Re^n$. An optimization problem is feasible if there is at least one point that satisfies the constraints of the problem, which is called a feasible point. The set of feasible points is called the feasible set or set of constraints. A vector $x*$ is called (total) Optimal or otherwise the solution of the problem A.2, if it has the lowest objective value of all the vectors in Ω that satisfy the constraints (achievable points), i.e. if for every $z \in \Omega$ with $g_i(z) \leq 0$ and $h_j(z) = 0$ for $i \in [1, m]$ and $j \in [1, p]$ holds that $f(x*) \leq f(z)$. Additionally, a vector x^+ is called a local optimal if there is $\epsilon > 0$ such that $f(x^+) \leq f(z)$ for the achievable z with $\|x^+ - z\|_2 \leq \epsilon$. If for a feasible point x, $g_i(x) = 0$, then this inequality constraint is called active in x, otherwise the inequality constraint $g_i(x) \leq 0$ is called inactive. Finally, it is stated that an optimization problem can be transformed into a problem form minimization A.2, if the opposite objective function $-f$ is considered.

A.1.2 Basic Definitions

An optimization problem is called a linear program, if the objective function and the constraint functions are linear functions of x. If some of the above functions are non linear, the problem is called non-linear problem. Additionally, if the set Ω contains integer sets, the problem is called an integer problem. A special category of non linear problems are convex optimization problems, in which for a convex set Ω, the objective function and the inequality constraint functions are convex, while the equality constraint functions are affine. In a convex optimization problem, the feasible set it is convex. Finally, for the opposite function $-f$, which is concave, the equivalent concave maximization problem derives.

Definition 1 *A set $\Omega \subset \Re^n$ is convex, if $\forall x, y \in \Omega$ and for every $\theta \in [0, 1]$, it holds*

$$\theta x + (1 - \theta) y \in \Omega \tag{A.3}$$

That is, for a convex set Ω, any straight line between two points of Ω is inside Ω.

Definition 2 *Given a convex set $\Omega \subseteq \Re^n$, a function $f : \Re^n \to R$ is*

- *convex in Ω, if*

$$f(ax + (1-a)y) \leq af(x) + (1-a)f(y), \forall x, y \in \Omega \quad \textit{and} \quad a \in (0,1) \tag{A.4}$$

- *strictly convex in Ω, if*

$$f(ax+(1-a)y) \leq af(x)+(1-a)f(y), \forall x, y(x \neq y) \in \Omega \quad \textit{and} \quad a \in (0,1) \tag{A.5}$$

- *strongly convex in Ω, if exist $c > 0$, s.t.*

$$f(ax+(1-a)y) \leq af(x)+(1-a)f(y)-\frac{c}{2}a(1-a)\|x-y\|^2, \forall x, y \in \Omega \quad \textit{and} \quad a \in (0,1) \tag{A.6}$$

That is, for a convex function, the straight line joining two points of the graph representation of the function is located above its graph. A function is called concave if $-f$ is convex. From the previous definition, it is obvious that the following are valid:

$\boxed{\text{Strongly Convex} \Rightarrow \text{Strictly convex} \Rightarrow \text{Convex}}$

Additionally, the curvature of a function can be verified with the help of the following theorem:

Theorem 1 *(Boyd et al., 2004). A differential function $f : \Re^n \to \Re$ is convex on the convex $\Omega \subseteq \Re^n$, iff one of the following conditions holds for $x, y \in \Omega$:*

First Degree Condition: $f(y) \geq f(x) + \nabla f(x)^T(y - x)$

Second degree condition (for twice differentiable f): $\nabla^2 f(x) \geq 0$

Note that $\nabla^2 f(x)$ represents the Hessian matrix of f, which is given as follows:

$$\mathcal{H}_f(x) = \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \ddots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \ddots & \ddots & \\ & & \ddots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \ddots & \frac{\partial^2 f}{\partial x_n} \end{bmatrix} \tag{A.7}$$

$\nabla^2 f() \geq 0$ means that the matrix $\nabla^2 f()$ is positive semi-definite. An important property of convex functions is the satisfaction of Jensen inequality. According to this inequality, if a function f is convex and the parameter x has some random distribution in Ω, then the following inequality holds:

$$f(\mathcal{E}[x]) \leq \mathcal{E}[f(x)] \tag{A.8}$$

, where $\mathcal{E}[.]$ is the expected value function. Since in some cases the curvature of a function is not necessary, the following definitions of functions are presented.

Definition 3 *A function $f : \Re^n \to \Re$ is called quasi-convex in the convex Ω, if for every $x, y \in \Omega$, the next inequality holds:*

$$f(ax + (1-a)y) \leq max\{f(x), f(y)\}, \forall a \in (0,1) \tag{A.9}$$

That is, for a quasi-convex function, the value of the function in a segment between two points does not exceed the maximum value of the two extreme points. A function is called quasi-concave, if $-f$ is quasi-convex. Every convex function is quasi-convex.

Definition 4 *A differenciable function $f : \Re^n \to \Re$ is called psedo-convex in the open $\Omega \subseteq \Re^n$, if for every $x, y \in \Omega$, with $\nabla f(x)^T(y-x) \geq 0$), holds $f(y) \geq f(x)$ (or if $f(y) < f(x), \nabla f(x)^T(y-x) < 0$).*

A function is called pseudo-concave if $-f$ is pseudo-concave. Any differential convex function is pseudo-convex.

A.1.3 Optimization Conditions

This subsection presents the basic optimization conditions of a point, for optimization problems with or without restrictions.

Optimization problems without restrictions

First, a subset of the optimization conditions for an optimization problem without restrictions is briefly presented A.1.

- Necessary Conditions

  1. First degree: if $f(x)$ is differentiable at $\bar{x}$ and $\bar{x}$ is a local optimum, then $\nabla f(\bar{x}) = 0$.
  2. Second degree: if $f(x)$ is twice differentiable at $\bar{x}$ and $\bar{x}$ is a local optimum, then $\nabla f(\bar{x}) = 0$ and $\nabla^2 f(\bar{x}) \geq 0$.

  If a point $\bar{x}$ satisfies $\nabla f(x) = 0$, this does not necessarily mean that it is an optimal point, but it is a stationary point. A stationary point can be an optimal point or a saddle point. For this reason, in addition to the necessary conditions, there are also the sufficient conditions that ensure the optimality of a point.

- Sufficient Conditions

  1. First degree: if $f(x)$ is quasi-convex at $\bar{x}$ and $\bar{x}$ is a local optimum, then $\nabla f(\bar{x}) = 0$.
  2. Second degree: if $f(x)$ is twice differentiable at $\bar{x}$, if $\nabla f(\bar{x}) = 0$ and $\nabla^2 f(\bar{x} \geq 0)$, then $\bar{x}$ is a local optimum.

Optimization problems with restrictions

The following describes some necessary and sufficient conditions for the optimization of the initial constraint optimization problem. More specifically, for A.2, which is not necessarily convex, the Lagrange function is defined as $\mathcal{L} : \Re^n \times \Re^m \times \Re^p \to \Re$

$$\mathcal{L} = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \tag{A.10}$$

where the vectors $\lambda = (\lambda_1, ..., \lambda_m)$ and $\mu = (\mu_1, ..., \mu_p)$ are called Lagrange multiplier vectors related to the constraints of equalities and inequalities, respectively. Next, the theorems that offer some necessary and capable conditions of first degree

optimization are presented. Regarding the conditions of second class optimization, the reader is referred to (Bertsekas, 2016).

Theorem 2 *(Necessary conditions Karush-Kuhn-Tucker ((Bazaraa et al., 2013)). Let the problem A.2 for open $\Omega \subset \Re^n$ with f, g_i for $i \in [1, m]$ and h_j for $j \in [1, p]$ be constantly differenciable in a random $x^* \in \Omega$. Additionally, let for the point x^* to hold suitable regularity conditions (constraint qualifications). Then, if x^* solves locally problem A.2, then scalar vectors $\lambda = (\lambda_1^*, ..., \lambda_m^*)$ and $\mu = (\mu_1^*, ..., \mu_p^*)$ exist, s.t.*

$$\begin{aligned} h_j(x^*) &= 0, \forall j \in [1, p] \\ g_i(x^*) &\leq 0, \forall i \in [1, m] \\ \lambda^* &\geq 0, \forall i \in [1, m] \\ \lambda_i^* g_i(x^*) &= 0, \forall i \in [1, m] \\ \nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \mu_j^* \nabla h_i(x^*) &= 0 \end{aligned} \tag{A.11}$$

These conditions are called the Karush-Kuhn-Tucker (KKT) conditions and the point which satisfies them is called the KKT point.

The first two conditions are the feasibility conditions of the initial problem, the third is the feasibility condition of the binary problem (Boyd and Vandenberghe, 2011), the fourth is referred to as complementary slackness, and the latter as stationarity condition. KKT conditions are necessary and not always sufficient conditions for solving the Optimization problem. Therefore, the solutions resulting from the KKT conditions are not necessary and solutions to the optimization problem, but they are a stationary point. However, if the optimization problem is convex, then the above conditions are also capable conditions. In addition, the following general theorem applies.

Theorem 3 *Let the problem A.2 for open $\Omega \subset \Re^n$ with f, g_i for $i \in [1, m]$ and h_j for $j \in [1, p]$ be constantly differenciable in a random $x^* \in \Omega$, for which the KTT conditions apply, meaning that there are scalar vectors λ^* and μ^*, s.t. A.11 applies.*

If f is psedo-convex at x^*, g_i are quasi-convex at x^* for $i \in [1, m]$ and h_j are quasi-convex at x^* for the j for which $\mu_j^* > 0$ and quasi-convex at x^* for the j for which $\mu_j^* < 0$, then point x^* is total optimum solution of A.2. If the conditions of curvature are limited to a region around x^*, then the x^* is a local minimum of A.2.

In a simpler form the above sentence applies to a convex set, with f pseudo-convex, g_i quasi-convex and h_j affine. Therefore, the theorem also applies to a convex problem. Finally, it is stated that an important feature of convex optimization problems is that each local optimal solution is also a total Optimum. Additionally, a feasible point x is optimal for a convex problem with a differential objective function, iff

$$\nabla f(x)^T (y - x) \geq 0 \tag{A.12}$$

for each feasible y, while for a convex problem without restrictions, the relative condition of the optimum x is

$$\nabla f(x) = 0 \tag{A.13}$$

A.1.4 Euler-Lagrange equation

A particularly useful equation for optimization problems that exhibits integrals in object functions and constraint functions is the Euler Lagrange equation (Zeidler, 2008), which is derived from the theory of calculus of variations. First, the following overview is presented concerning the Basic Optimization Problem from which the Euler-Lagrange equation is derived, and then some generalizations of the equation are presented.

Theorem 4 *(Zeidler, 2008) Let $t_a, t_b, g(t_a), q(t_b) \in Re$, with $t_a < t_b$ and the following optimization problem:*

$$\min_{q(t)} \int_{t_a}^{t_b} F(q(t), g'(t), t)dt, with q(t_a) = q_a, q(t_b) = q_b \tag{A.14}$$

, where function $F : \Re \times \Re \times [t_a, t_b] \rightarrow \Re$ is twice differenciable. If $q(t)$, with $t \in [t_a, t_b]$ is twice differenciable and solves problem A.14, then the following equation holds (Euler-Lagrange equation):

$$\frac{d}{dt}\frac{\partial F(q(t), q'(t), t)}{\partial q'} - \frac{\partial F(q(t), q'(t), t)}{\partial q} = 0 \tag{A.15}$$

The solution of the Euler-Lagrange equation gives the stationary points of the problem in result to be the necessary condition for the solution of the problem. For competent conditions the reader is referred to (Zeidler, 2008). It is stated that in the special case where $F(q(t), t)$, the Euler-Lagrange equation is simplified to the following equation and in addition there is no requirement for border conditions:

$$\frac{\partial F(q(t), t)}{\partial q} = 0 \tag{A.16}$$

Additionally, in a more general case where $t = (t_1, ..., T_N)$ is a vector of values in the range $D \in \Re^N$ and $q = (q_1, ..., q_K) = q(t)$ is a vector function under specific boundary conditions, the following system of Euler-Lagrange equations arises:

$$\sum_{n=1}^{N} \frac{\partial}{\partial t_n} \frac{\partial F}{\partial \frac{\partial q_k}{\partial t_n}} - \frac{\partial F}{\partial q_k} = 0, for \quad k \in [1, K] \tag{A.17}$$

, where $F = F(q_1, ..., q_K, \frac{\partial q_1}{\partial t_1}, ..., \frac{\partial q_1}{\partial t_N}, ..., \frac{\partial q_K}{\partial t_1}, ..., \frac{\partial q_K}{\partial t_N}, t_1, ..., t_N)$.

Finally, it is reported that for the problems where constraints (with or without integrals) apply, a function with Lagrange multipliers respectively with the Lagrange function of the Optimization Theory is defined, which is introduced in the Euler-Lagrange equation instead of F (Zeidler, 2008). Therefore, the KKT conditions can be used in combination with the Euler-Lagrange equation for optimization problems with limitations that include integrals.

A.2 Inequalities - Variation Theory

This section presents the inequalities variation theory, which deals with a more generic class of problems in non linear analysis. More specifically, the Inequalities Variation Theory (Scutari et al., 2010) is suitable for the study and solution of a wide range of mathematical problems, such as computer systems, computer graphs, completeness problems, fixed point problems etc. It is a powerful Mathematical tool for studying the problems of interactions between different entities, which is common in edge networks and can be applied even in cases where the classic game theory does not apply.

Next, some basic definitions and theorems of the inequalities changes theory are presented. Next, the existence and uniqueness of the solution of a change inequality problem is studied. Finally, some basic elements from the game theory and more specifically the cooperative game theory are presented, as well as its connection with the theory of changes inequalities.

Basic definitions

The inequalities change theory is a general mathematical framework that encompasses convex optimization and is directly related to the game theory. More specifically, the problem of Variational Inequality (VI) is defined below.

Definition 5 *Given a set $K \subseteq R^n$ and a function $F : K \to R^n$, the problem VI, which is defined as $VI(K, F)$, aims to find an $x^* \in K$, called solution of VI, s.t.*

$$F(x^*)^T(x - x^*) \geq 0, \forall x \in K \tag{A.18}$$

, where $(.)^T$ is the transpose matrix, or

$$\langle F((x^*), x - x^*)\rangle \geq 0, \forall x \in K \tag{A.19}$$

where $\langle ., . \rangle$ is the representation of the inner product. The set of the the solutions

to this problem are denoted by $SOL(K, F)$.

Next, it is assumed that K is closed and F is continuous. A problem of variation inequality is that it can be mapped to a convex optimization problem A.2, where ∇f is replaced by a general function F (A.12), as shown in the following sentence.

Statement 1 *(Nagurney, 1999). Let x^* be the solution of the following optimization problem.*

$$\begin{aligned} &\min_x f(x) \\ &s.t. \quad x \in \mathcal{K} \end{aligned} \tag{A.20}$$

, where $f : \mathcal{K} \to \Re$ is continuously differentiable and the set $\mathcal{K} \subseteq \Re^n$ is closed and convex. Then x^* is a solution of problem $VI(\mathcal{K}, \nabla f)$.

If f is a convex function then the inverse is also true, while for $\mathcal{K} = R^n$ the problem becomes an optimization problem without restrictions. Note that, if the function F cannot be expressed as the gradient of a potential function f (i.e. when F does not have a symmetric Jacobian array), VI problems are separated from the classical convex optimization problems, making the variation inequalities theory more general, since it includes a wider range of problems.

In addition, the following suggestions apply.

Statement 2 *(Nagurney, 1999). The problem of solving a system of equations $F(x) = 0$ is equivalent to problem $VI(\Re^n, F)$.*

Statement 3 *(Nagurney, 1999). Let's be a Nonlinear Complementarity Problem (NCP), which for a function $F : \Re^n_+ \to \Re^n$ is defined as the problem of finding a $x \in \Re^n$ such that*

$$0 \leq x \perp F(x) \geq 0 \tag{A.21}$$

The above NCP problem (F) is equivalent to problem $VI(\Re^n, F)$.

Note that for $z, y \in \Re^n, 0 \leq z \perp y \geq 0$ means that $z \geq 0, y \geq 0$ and $z^T y = 0$ (Orthogonality).

A.2.1 KKT conditions

For problems in VI class, the KKT conditions are defined accordingly. More specifically, let problem $VI(\mathcal{K}, F)$ and set $\mathcal{K}$ to incorporate constraints on inequalities and equalities as follows:

$$\mathcal{K} \equiv \{x \in \Re^n : h(x) = 0, g(x) \leq 0\} \tag{A.22}$$

, where $h : \Re^n \to \Re^p$ and $g : \Re^n \to \Re^m$ be constantly differenciable. Also, let some suitable conditions of regularity apply (Facchinei and Pang, 2003). Then the KKT conditions for the specific problem are the following:

$$\begin{aligned} h(x) &= 0 \\ 0 \leq \lambda \perp g(x) &\leq 0 \\ F(x) + \lambda^T \nabla g(x) + \mu^T \nabla h(x) &= 0 \end{aligned} \tag{A.23}$$

Additionally, the following theorem applies:

Theorem 5 *(Facchinei and Pang, 2003) Let $\mathcal{K} \equiv \{x \in \Re^n : h(x) = 0, g(x) \leq 0\}$ with $h : \Re^n \to \Re^p$ and $g : \Re^n \to \Re^m$ to be constantly differenciable and a function $F : \mathcal{K} \to \Re^n$. The following applies:*

- *Let $x \in SOL(\mathcal{K}, F)$. If the Abadie normality conditions (Facchinei and Pang, 2003) apply to x, then there exist vectors $\mu \in \Re^p$ and $\lambda \in \Re^m$ s.t. the conditions KKT A.23 apply.*
- *Conversely, if h is affine, g is convex and (x, λ, μ) satisfies KKT A.23, then it is also a solution of $VI(\mathcal{K}, F)$.*

A.2.2 Existence and Uniqueness of the Solution

Next, the existence and uniqueness of the solutions of a VI problem are studied. More specifically, regarding the existence of solutions, the following statement applies.

Theorem 6 *(Facchinei and Pang, 2003) Let a convex and solid set $K \subseteq \Re^n$ and a continuous function $F : \mathcal{K} \to \Re^n$. The set $SOL(\mathcal{K}, F)$ is non empty and solid.*

However, under certain conditions for the function F, a problem VI can have solutions without the set $\mathcal{K}$ being bounded. First, some definitions are given for F's monotony, which interacts with the corresponding curvature of f in the classical optimization problems.

Definition 6 *Given a convex set $\mathcal{K} \subseteq \Re^n$, a function $F : \mathcal{K} \to \Re^n$ is:*

- *monotone in $\mathcal{K}$, if:*

$$(F(x) - F(x)^T)(x - y) \geq 0, \forall x, y \in \mathcal{K} \tag{A.24}$$

- *strictly monotone in $\mathcal{K}$, if:*

$$(F(x) - F(x)^T)(x - y) \geq 0, \forall x, y \in \mathcal{K} \quad \textit{and} \quad x \neq y \tag{A.25}$$

- *strongly monotone in $\mathcal{K}$, if:*

$$(F(x) - F(x)^T)(x - y) \geq c \left\| x - y \right\|^2, \forall x, y \in \mathcal{K} \tag{A.26}$$

References

Aderaldo, C. M., Mendonça, N. C., Schmerl, B., and Garlan, D. (2019). Kubow: An architecture-based self-adaptation service for cloud native applications. In *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, volume 2, pages 42–45. ACM.

Adhikari, S., Dunn, T., and Hsiao, E. (2016). Augmented reality system for product identification and promotion. US Patent 9,286,721.

Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face recognition with local binary patterns. In *European conference on computer vision*, pages 469–481. Springer.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1988). Network flows.

Alnoman, A., Sharma, S. K., Ejaz, W., and Anpalagan, A. (2019). Emerging edge computing technologies for distributed iot systems. *IEEE Network*, 33(6):140–147.

Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'keeffe, D., Stillwell, M. L., et al. (2016). {SCONE}: Secure linux containers with intel {SGX}. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 689–703.

Athanesious, J. J. and Suresh, P. (2012). Systematic survey on object tracking methods in video. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(8):242–247.

Babu, S. A., Hareesh, M., Martin, J. P., Cherian, S., and Sastri, Y. (2014). System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver. In *2014 fourth international conference on advances in computing and communications*, pages 247–250. IEEE.

Baltieri, D., Vezzani, R., Cucchiara, R., Utasi, A., Benedek, C., and Szirányi, T. (2011). Multi-view people surveillance using 3d information. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1817–1824. IEEE.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2013). *Nonlinear programming: theory and algorithms*. John Wiley & Sons.

Beal, L. D., Hill, D. C., Martin, R. A., and Hedengren, J. D. (2018). Gekko optimization suite. *Processes*, 6(8):106.

Bertsekas, D. P. (2016). *Nonlinear programming*. Athena scientific.

Besse, C. and Chaib-draa, B. (2007). An Efficient Model for Dynamic and Constrained Resource Allocation Problems. In *2nd COPLAS '07*.

Blog, G. A. (2018). MobileNetV2: The Next Generation of On-Device Computer Vision Networks.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Boyd, S. P. and Vandenberghe, L. (2011). *Convex optimization*. Cambridge Univ. Pr.

Brogi, A. and Forti, S. (2017). Qos-aware deployment of iot applications through the fog. *IEEE Internet of Things Journal*, 4(5):1185–1192.

Burns, B., Beda, J., Hightower, K., and Evenson, L. (2022). *Kubernetes: up and running*. " O'Reilly Media, Inc.".

Byrne, J., Svorobej, S., Giannoutakis, K. M., Tzovaras, D., Byrne, P. J., Östberg, P.-O., Gourinovitch, A., and Lynn, T. (2017). A review of cloud computing simulation platforms and related environments. In *International Conference on Cloud Computing and Services Science*, volume 2, pages 679–691. SciTePress.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

Carvalho, G., Cabral, B., Pereira, V., and Bernardino, J. (2021). Edge computing: current trends, research challenges and future directions. *Computing*, 103(5):993–1023.

Chen, M., Li, W., Fortino, G., Hao, Y., Hu, L., and Humar, I. (2019). A dynamic service migration mechanism in edge cognitive computing. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–15.

Chen, N. and Chen, Y. (2018). Smart City Surveillance at the Network Edge in the Era of IoT: Opportunities and Challenges. In *Smart Cities*, pages 153–176.

Chen, N., Chen, Y., Ye, X., Ling, H., Song, S., and Huang, C.-T. (2017). Smart city surveillance in fog computing. In *Advances in mobile cloud computing and big data in the 5G era*, pages 203–226. Springer.

Chen, N., Chen, Y., You, Y., Ling, H., Liang, P., and Zimmermann, R. (2016). Dynamic urban surveillance video stream processing using fog computing. In *2016 IEEE second international conference on multimedia big data (BigMM)*, pages 105–112. IEEE.

Chen, S. and Zhou, M. (2021). Evolving container to unikernel for edge computing and applications in process industry. *Processes*, 9(2):351.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Corso, J. J., Alahi, A., Grauman, K., Hager, G. D., Morency, L.-P., Sawhney, H., and Sheikh, Y. (2016). Video analysis for body-worn cameras in law enforcement. *arXiv preprint arXiv:1604.03130*.

Costa, D. G., Figuerêdo, S., and Oliveira, G. (2017). Cryptography in wireless multimedia sensor networks: A survey and research directions. *Cryptography*, 1(1):4.

D'Angelo, G., Ferretti, S., and Ghini, V. (2017). Modeling the internet of things: a simulation perspective. In *2017 International Conference on High Performance Computing & Simulation (HPCS)*, pages 18–27. IEEE.

Das, A., Patterson, S., and Wittie, M. (2018). Edgebench: Benchmarking edge computing platforms. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 175–180. IEEE.

Dautov, R., Distefano, S., Bruneo, D., Longo, F., Merlino, G., Puliafito, A., and Buyya, R. (2018). Metropolitan intelligent surveillance systems for urban areas by harnessing iot and edge computing paradigms. *Software: Practice and experience*, 48(8):1475–1492.

Dayalan, U. K., Fezeu, R. A., Varyani, N., Salo, T. J., and Zhang, Z.-L. (2021). Veeredge: Towards an edge-centric iot gateway. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 690–695. IEEE.

Debab, R. and Hidouci, W. K. (2020). Containers runtimes war: a comparative study. In *Proceedings of the Future Technologies Conference*, pages 135–161. Springer.

Demosthenous, G. and Vassiliades, V. (2021). Continual learning on the edge with tensorflow lite. *arXiv preprint arXiv:2105.01946*.

Dhar, S. and Bose, I. (2021). Securing iot devices using zero trust and blockchain. *Journal of Organizational Computing and Electronic Commerce*, 31(1):18–34.

Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611.

Doan, T. V., Fan, Z., Nguyen, G. T., Salah, H., You, D., and Fitzek, F. H. (2020). Follow me, if you can: A framework for seamless migration in mobile edge cloud. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1178–1183. IEEE.

Doan, T. V., Nguyen, G. T., Reisslein, M., and Fitzek, F. H. (2021). Fast: Flexible and low-latency state transfer in mobile edge computing. *IEEE Access*, 9:115315–115334.

Du, R., Bista, S., and Varshney, A. (2016). Video fields: fusing multiple surveillance videos into a dynamic virtual environment. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 165–172.

Elgammal, A. (2014). Background subtraction: Theory and practice. *Synthesis Lectures on Computer Vision*, 5(1):1–83.

Facchinei, F. and Pang, J.-S. (2003). *Finite-dimensional variational inequalities and complementarity problems*. Springer.

Farris, I., Taleb, T., Flinck, H., and Iera, A. (2018). Providing ultra-short latency to user-centric 5g applications at the mobile network edge. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3169.

Fienen, M. N. and Plant, N. G. (2015). A cross-validation package driving netica with python. *Environmental Modelling & Software*, 63:14–23.

Floudas, C. A., . P. P. M. (2009). Encyclopedia of optimization (2nd ed.). Springer.

Gallagher, J. C., Boddhu, S. K., and Vigraham, S. (2005). A reconfigurable continuous time recurrent neural network for evolvable hardware applications. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2461–2468. IEEE.

Galteri, L., Bertini, M., Seidenari, L., and Del Bimbo, A. (2018). Video compression for object detection algorithms. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3007–3012. IEEE.

García-Pérez, C. A. and Merino, P. (2018). Experimental evaluation of fog computing techniques to reduce latency in lte networks. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3201.

Giust, F., Costa-Perez, X., and Reznik, A. (2017). Multi-access edge computing: An overview of etsi mec isg. *IEEE 5G Tech Focus*, 1(4):4.

Gouareb, R., Friderikos, V., and Aghvami, A.-H. (2018). Virtual network functions routing and placement for edge cloud latency minimization. *IEEE Journal on Selected Areas in Communications*, 36(10):2346–2357.

Gries, P., Campbell, J., and Montojo, J. (2017). *Practical programming: an introduction to computer science using Python 3.6*. Pragmatic Bookshelf.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48.

Ha, K., Abe, Y., Eiszler, T., Chen, Z., Hu, W., Amos, B., Upadhyaya, R., Pillai, P., and Satyanarayanan, M. (2017). You can teach elephants to dance: Agile vm handoff for edge computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–14.

HajiRassouliha, A., Taberner, A. J., Nash, M. P., and Nielsen, P. M. (2018). Suitability of recent hardware accelerators (dsps, fpgas, and gpus) for computer vision and image processing algorithms. *Signal Processing: Image Communication*, 68:101–119.

Hall, B. and Trivedi, M. (2002). A novel graphical interface and context aware map for incident detection and monitoring. In *9th World Congress on Intelligent Transport Systems*. Citeseer.

Halpern, J. and Pignataro, C. (2015). RFC 7665 - Service Function Chaining (SFC) Architecture.

He, D., Wang, Z., and Liu, J. (2018). A survey to predict the trend of ai-able server evolution in the cloud. *IEEE Access*, 6:10591–10602.

He, T., Toosi, A. N., and Buyya, R. (2021). Efficient large-scale multiple migration planning and scheduling in sdn-enabled edge computing. *arXiv preprint arXiv:2111.08936*.

Hedengren, J., Mojica, J., Cole, W., and Edgar, T. (2012). Apopt: Minlp solver for differential and algebraic systems with benchmark testing. In *Proceedings of the INFORMS National Meeting, Phoenix, AZ, USA*, volume 1417, page 47.

Hinton, G. E., Sejnowski, T. J., et al. (1986). Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2.

Hossain, M. S. (2011). Qos-aware service composition for video surveillance. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–5.

Hossain, M. S., Hassan, M. M., Al Qurishi, M., and Alghamdi, A. (2012). Resource allocation for service composition in cloud-based video surveillance platform. In *2012 IEEE international conference on multimedia and expo workshops*, pages 408–412. IEEE.

Hossain, S. and Lee, D.-j. (2019). Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices. *Sensors*, 19(15):3371.

Ichinose, A., Takefusa, A., Nakada, H., and Oguchi, M. (2017). A study of a video analysis framework using kafka and spark streaming. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2396–2401. IEEE.

Jain, A., Awan, A. A., Anthony, Q., Subramoni, H., and Panda, D. K. D. (2019). Performance characterization of dnn training using tensorflow and pytorch on modern clusters. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11. IEEE.

Jin, Z., Zhu, Y., Zhu, J., Yu, D., Li, C., Chen, R., Akkus, I. E., and Xu, Y. (2021). Lessons learned from migrating complex stateful applications onto serverless platforms. In *Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 89–96.

Kamgar-Parsi, B., Lawson, W., and Kamgar-Parsi, B. (2011). Toward development of a face recognition system for watchlist surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1925–1937.

Kayal, P. (2020). Kubernetes: Towards deployment of distributed iot applications in fog computing. In *Companion of the ACM/SPEC International Conference on Performance Engineering*, pages 32–33.

Kecskemeti, G., Casale, G., Jha, D. N., Lyon, J., and Ranjan, R. (2017). Modelling and simulation challenges in internet of things. *IEEE cloud computing*, 4(1):62–69.

Khan, A., Nawaz, U., Ulhaq, A., and Robinson, R. W. (2020). Real-time plant health assessment via implementing cloud-based scalable transfer learning on aws deeplens. *Plos one*, 15(12):e0243243.

Khan, S. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265.

Khochare, A., Ravindra, P., Reddy, S. P., and Simmhan, Y. (2017). Distributed video analytics across edge and cloud using echo. In *International Conference on Service-Oriented Computing*, pages 402–407. Springer.

Kim, H.-S. (2016). Fog computing and the internet of things: Extend the cloud to where the things are. *International Journal of Cisco*.

Kitchin, R. (2014). The real-time city? Big data and smart urbanism. *GeoJournal*, 79(1):1–14.

Kliazovich, D., Bouvry, P., and Khan, S. U. (2012). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

Kumar, R., Sawhney, H., Samarasekera, S., Hsu, S., Tao, H., Guo, Y., Hanna, K., Pope, A., Wildes, R., Hirvonen, D., et al. (2001). Aerial video surveillance and exploitation. *Proceedings of the IEEE*, 89(10):1518–1539.

Kumar, R., Sawhney, H. S., Guo, Y., Hsu, S., and Samarasekera, S. (2000). 3d manipulation of motion imagery. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 1, pages 17–20. IEEE.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018). Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250.

Lantagne, M., Parizeau, M., and Bergevin, R. (2003). Vip: Vision tool for comparing images of people. In *Vision Interface*, volume 2.

Lee, H., Oh, J., Kim, K., and Yeon, H. (2017). A data streaming performance evaluation using resource constrained edge device. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 628–633. IEEE.

Lenc, L. and Král, P. (2016). Local binary pattern based face recognition with automatically detected fiducial points. *Integrated Computer-Aided Engineering*, 23(2):129–139.

Li, C., Wang, Y., Tang, H., and Luo, Y. (2019). Dynamic multi-objective optimized replica placement and migration strategies for saas applications in edge cloud. *Future Generation Computer Systems*, 100:921–937.

Li, J., Jin, J., Yuan, D., and Zhang, H. (2017). Virtual fog: A virtualization enabled fog computing framework for internet of things. *IEEE Internet of Things Journal*, 5(1):121–131.

Li, Q.-q., Zhang, T., and Yu, Y. (2011). Using cloud computing to process intensive floating car data for urban traffic surveillance. *International Journal of Geographical Information Science*, 25(8):1303–1322.

Lima, S., Rocha, A., and Roque, L. (2019). An overview of openstack architecture: a message queuing services node. *Cluster Computing*, 22(3):7087–7098.

Makaratzis, A. T., Giannoutakis, K. M., and Tzovaras, D. (2018). Energy modeling in cloud simulation frameworks. *Future Generation Computer Systems*, 79:715–725.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations research*, 8(2):219–223.

Mansouri, Y. and Babar, M. A. (2021). A review of edge computing: Features and resource virtualization. *Journal of Parallel and Distributed Computing*, 150:155–183.

Mendu, M., Krishna, B., Sandeep, C., Padmaja, C., and Sultana, F. (2022). Creating and running apps on the hybrid cloud of microsoft azure. In *AIP Conference Proceedings*, volume 2418, page 020039. AIP Publishing LLC.

Meng, X. and Lu, W. (2021). Container-based fast service migration method for mobile edge computing. *Journal of Circuits, Systems and Computers*, 30(15):2250117.

Miao, Y., Lyu, F., Wu, F., Wu, H., Ren, J., Zhang, Y., and Shen, X. S. (2022). Mobility-aware service migration for seamless provision: A reinforcement learning approach. In *ICC 2022-IEEE International Conference on Communications*, pages 5064–5069. IEEE.

Morabito, R., Farris, I., Iera, A., and Taleb, T. (2017). Evaluating performance of containerized iot services for clustered devices at the network edge. *IEEE Internet of Things Journal*, 4(4):1019–1030.

Nagurney, A. (1999). Network economics: A variational inequality approach, kluwer academic. *Dordrecht, The Netherlands*.

Nasir, M., Muhammad, K., Lloret, J., Sangaiah, A. K., and Sajjad, M. (2019). Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. *Journal of Parallel and Distributed Computing*, 126:161–170.

Neil, D., Pfeiffer, M., and Liu, S.-C. (2016). Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29.

Núñez, A., Vázquez-Poletti, J. L., Caminero, A. C., Castañé, G. G., Carretero, J., and Llorente, I. M. (2012). icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209.

Nwokolo, C. P. and Inyiama, H. C. (2017). Quality of service evaluation in on-demand cloud-based video surveillance. In *2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON)*, pages 532–537.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., Mukherjee, S., Aggarwal, J., Lee, H., Davis, L., et al. (2011). A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE.

Orsini, G., Bade, D., and Lamersdorf, W. (2018). Cloudaware: empowering context-aware self-adaptation for mobile applications. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3210.

Pacheco, J. and Hariri, S. (2018). Anomaly behavior analysis for iot sensors. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3188.

Pérez Castillo, Ò. (2021). Development of a virtualization framework with lxd. B.S. thesis, Universitat Politècnica de Catalunya.

Piccardi, M. (2004). Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, pages 3099–3104.

Porikli, F., Bremond, F., Dockstader, S. L., Ferryman, J., Hoogs, A., Lovell, B. C., Pankanti, S., Rinner, B., Tu, P., and Venetianer, P. L. (2013). Video surveillance: past, present, and now the future [dsp forum]. *IEEE Signal Processing Magazine*, 30(3):190–198.

Prabhakar, N. and Anbarasi, L. J. (2021). Exploration of the global air transport network using social network analysis. *Social Network Analysis and Mining*, 11(1):1–12.

Prati, A., Shan, C., and Wang, K. I.-K. (2019). Sensors, vision and networks: From video surveillance to activity recognition and health monitoring. *Journal of Ambient Intelligence and Smart Environments*, 11(1):5–22.

Qiu, Y., Niu, Z., Song, B., Ma, T., Al-Dhelaan, A., and Al-Dhelaan, M. (2022). A novel generative model for face privacy protection in video surveillance with utility maintenance. *Applied Sciences*, 12(14):6962.

Randazzo, A. and Tinnirello, I. (2019). Kata containers: An emerging architecture for enabling mec services in fast and secure way. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 209–214. IEEE.

Ranzato, M. (2014). Video (language) modeling: A baseline for generative models of natural videos. *arXiv:1412.6604*.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.

Ray, K. and Banerjee, A. (2021). Horizontal auto-scaling for multi-access edge computing using safe reinforcement learning. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(6):1–33.

Ray, K., Banerjee, A., and Narendra, N. C. (2020). Proactive microservice placement and migration for mobile edge computing. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 28–41. IEEE.

Rodrigues, T. G., Suto, K., Nishiyama, H., and Kato, N. (2016). Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control. *IEEE Transactions on Computers*, 66(5):810–819.

Rodríguez-Silva, D. A., Adkinson-Orellana, L., Gonz'lez-Castano, F., Armino-Franco, I., and Gonz'lez-Martinez, D. (2012). Video surveillance based on cloud storage. In *2012 IEEE fifth international conference on Cloud computing*, pages 991–992. IEEE.

Rosen, R. (2016). Namespaces and cgroups, the basis of linux containers. *Seville, Spain, Feb.*

Samaniego, M. and Deters, R. (2018). Zero-trust hierarchical management in iot. In *2018 IEEE international congress on Internet of Things (ICIOT)*, pages 88–95. IEEE.

Satyanarayanan, M. (2019). The Emergence of Edge Computing. *Computer*, 50(1):30–39.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23.

Scutari, G., Palomar, D. P., Facchinei, F., and Pang, J.-S. (2010). Convex optimization, game theory, and variational inequality theory. *IEEE Signal Processing Magazine*, 27(3):35–49.

Sebe, I. O., Hu, J., You, S., and Neumann, U. (2003). 3d video surveillance with augmented virtual environments. In *First ACM SIGMM international workshop on Video surveillance*, pages 107–112.

Sharma, P. and Singh, A. (2017). Era of deep neural networks: A review. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646.

Skarlat, O., Nardelli, M., Schulte, S., and Dustdar, S. (2017). Towards qos-aware fog service placement. In *2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC)*, pages 89–96. IEEE.

Song, B., Tian, Y., and Zhou, B. (2014). Design and evaluation of remote video surveillance system on private cloud. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pages 256–262.

Sotiriadis, S., Bessis, N., Antonopoulos, N., and Anjum, A. (2013). Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 90–97. IEEE.

Sotiriadis, S., Bessis, N., Asimakopoulou, E., and Mustafee, N. (2014). Towards simulating the internet of things. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 444–448. IEEE.

Sun, F., Cheng, N., Zhang, S., Zhou, H., Gui, L., and Shen, X. (2018). Reinforcement learning based computation migration for vehicular cloud computing. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Sun, H., Shi, W., Liang, X., and Yu, Y. (2019). Vu: Edge computing-enabled video usefulness detection and its application in large-scale video surveillance systems. *IEEE Internet of Things Journal*, 7(2):800–817.

Sun, L., Li, Y., and Memon, R. A. (2017). An open iot framework based on microservices architecture. *China Communications*, 14(2):154–162.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681.

Tazaki, H., Moroo, A., Kuga, Y., and Nakamura, R. (2021). How to design a library os for practical containers? In *Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 15–28.

Thorp, K. R. and Bronson, K. F. (2013). A model-independent open-source geospatial tool for managing point-based environmental model simulations at multiple spatial locations. *Environmental modelling & software*, 50:25–36.

Torabi, A., Massé, G., and Bilodeau, G.-A. (2012). An iterative integrated framework for thermal–visible image registration, sensor fusion, and people tracking for video surveillance applications. *Computer Vision and Image Understanding*, 116(2):210–221.

Tsakanikas, V. and Dagiuklas, T. (2018). Video surveillance systems-current status and future trends. *Computers & Electrical Engineering*, 70:736–753.

Tsakanikas, V. and Dagiuklas, T. (2021). Enabling real-time ai edge video analytics. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE.

Tsakanikas, V. and Dagiuklas, T. (2022a). A generic framework for deploying video analytic services on the edge. *Transactions on Cloud Computing - R1 revision*.

Tsakanikas, V. and Dagiuklas, T. (2022b). Vfcsim: A simulation framework of real-time multi-service virtual function chains deployment. In *GLOBECOM 2022 - IEEE Global Communications Conference*. IEEE.

Tsushita, H. and Zin, T. T. (2018). A study on detection of abnormal behavior by a surveillance camera image. In *International Conference on Big Data Analysis and Deep Learning Applications*, pages 284–291. Springer.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

Wang, J., Pan, J., Esposito, F., Calyam, P., Yang, Z., and Mohapatra, P. (2019a). Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–23.

Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., and Leung, K. K. (2019b). Dynamic service migration in mobile edge computing based on markov decision process. *IEEE/ACM Transactions on Networking*, 27(3):1272–1288.

Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 24(1):34–58.

Yaseen, Q., Albalas, F., Jararwah, Y., and Al-Ayyoub, M. (2018). Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks. *Transactions on Emerging Telecommunications Technologies*, 29(4):e3183.

Yousafzai, A., Yaqoob, I., Imran, M., Gani, A., and Noor, R. M. (2019). Process migration-based computational offloading framework for iot-supported mobile edge/cloud computing. *IEEE internet of things journal*, 7(5):4171–4182.

Zafeiriou, S., Zhang, C., and Zhang, Z. (2015). A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, 138:1–24.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65.

Zeidler, E. (2008). *Oxford User's Guide to Mathematics*. Oxford Univ. Press.

Zeng, X., Garg, S. K., Strazdins, P., Jayaraman, P., Georgakopoulos, D., and Ranjan, R. (2016). Iotsim: a cloud based simulator for analysing iot applications. *arXiv preprint arXiv:1602.06488*.

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. (2017). Live video analytics at scale with approximation and {Delay-Tolerance}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392.

Zhang, Q., Yu, Z., Shi, W., and Zhong, H. (2016). Demo abstract: Evaps: Edge video analysis for public safety. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 121–122. IEEE.

Zhou, W., Saha, D., and Rangarajan, S. (2015). A system architecture to aggregate video surveillance data in smart cities. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

Zotkin, D. N., Duraiswami, R., and Davis, L. S. (2002). Joint audio-visual tracking using particle filters. *EURASIP Journal on Advances in Signal Processing*, 2002(11):1–11.

Zou, X. and Bhanu, B. (2005). Pixels that sound. In *CVPR'05: Proc. 2005 Conf. Computer Vision and Pattern Recognition (CVPR'05)*, pages 88–95.

Zou, Y., Shi, G., Jin, Y., and Zheng, Y. (2009). Extraocular image processing for retinal prosthesis based on dsp. In *2009 4th IEEE International Conference on Nano/Micro Engineered and Molecular Systems*, pages 563–566.