University College London

On the Deployment of Low Latency Network Applications over Third-Party In-Network Computing Resources

Doctor of Philosophy

in

Electronic and Electrical Engineering

by

Argyrios G. Tasiopoulos

2018

I, Argyrios Tasiopoulos, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Electronic & Electrical Engineering
University College London

Abstract

An increasing number of Low Latency Applications (LLAs) in the entertainment (Virtual/Augmented Reality), Internet-of-Things (IoT), and automotive domains require response times that challenge the traditional application provisioning process into distant data centres. At the same time, there is a trend in deploying In-Network Computing Resources (INCRs) *closer* to end users either in the form of network equipment, with capabilities of performing general-purpose computations, and/or in the form of commercial off-the-self "data centres in a box", *i.e.,* cloudlets, placed at different locations of Internet Service Providers (ISPs). That is, INCRs extend cloud computing at the edge and middle-tier locations of the network, providing significantly smaller response times than those achieved by the current "client-to-cloud" network model.

In this thesis, we argue about the necessity of exploiting INCRs for application provisioning with the purpose of improving LLAs' Quality of Service (QoS) by essentially deploying applications *closer* to end users. To this end, this thesis investigates the deployment of LLAs over INCRs under fixed, mobile, and disrupted user connectivity environments. In order to fully reap the benefits of INCRs, we develop for each connectivity scenario algorithmic frameworks that are centred around the concept of a market, where LLAs lease existing INCRs. The proposed frameworks take into account the particular characteristics of INCRs, such as their limited capacity in hosting application instances, and LLAs, by addressing the number of instances each application should deploy at each computing resource over time. Furthermore, since typically the smooth operation of network applications is supported by Network Functions, such as load balancers, firewalls *etc.*, we consider the deployment of complementary Virtual Network Functions for backing LLAs' provisioning over INCRs. Overall, the key goal of this thesis is the investigation of using an enhanced Internet through INCRs as the communication platform for LLAs.

Acknowledgements

I would like to express my gratitude to all those who made this dissertation possible, and first of all to my supervisor, Professor George Pavlou, for giving me the opportunity to pursue a PhD in such an inspiring and creative environment. I am grateful for his guidance and the freedom I had in following the research directions of my choice. I would also like to thank Dr. Ioannis Psaras for his help and the numerous discussions, over the last 4 years, that help me to evolve as a researcher.

I also want to express my appreciation to the people who help me start my PhD journey at first place. In particular my master thesis supervisor Dr. Stavros Toumpis, who encouraged me to follow this path, and Dr. Cecilia Mascolo, for her support on my first steps in UK.

I would like to thank my lab-mates for their support and for being there to share my problems, usually during a pub session, especially Dr. Onur Ascigil, Dr. Sergi Rene, Dr. Daphne Tuncer, Dr. Marinos Charalambides, Dr. Stuart Clayman, Dr. Truong Khoa Phan, Dr. Michal Krol, Dr. David Griffin, Dr. Konstantinos Katsaros, Dr. Wei Koong Chai, Gioacchino Tangari and of course my academic brother Binxu Yang. They made my last 4 years more enjoyable and memorable.

I also received a great amount of encouragement from many friends and ex-colleagues with whom I had the pleasure to share my time in London, especially Dr. Lorenzo Saino, Dr. Vasilis Sourlas, and Orestis Dikaios. I would also like to thank my friends from Greece, who were always there for me on my short trips back.

I also thank the examiners of this dissertation., Dr. Dimitrios Pezaros and Dr. Miguel Rodrigues, for their time as well as all the valuable feedback they provided.

Lastly, and most importantly, I am deeply grateful to my family and especially to my father George, my mother Angela, and my brother Panayiotis for their support and confidence in me.

Contents

List of Figures

List of Tables

Abbreviations

AP	Access Point
AppSP	Application Service Provider
BS	Base Stations
COTS	Commercial Off-The-Self
D2D	Device-to-Device
FCT	Flow Completion Time
INCP	In-Network Computing Providers
INCR	In-Network Computing Resource
IoT	Internet-of-Things
ISP	Internet Service Provider
LLA	Low Latency (Network) Application
NF	Network Function
NFI	Network Function Instance
NFV	Network Function Virtualisation
PUSH	Pull and Share
QoS	Quality of Service
SDN	Software Defined Network
SFC	Service Function Chaining
VED	Vickrey-English-Dutch
VM	Virtual Machine
VNF	Virtual Network Function

Publications

- Tasiopoulos, A.G., Ascigil, O., Psaras, I., Toumpis, S., and Pavlou, G., **On-path Cloudlet Pricing for Low Latency Application Provisioning.** To Appear in IEEE International Symposium on Local and Metropolitan Area Networks, 2018.

- Tasiopoulos, A.G., Ascigil, O., Psaras, I., and Pavlou, G., **Edge-MAP: Auction Markets for Edge Resource Provisioning.** To Appear in IEEE World of Wireless, Mobile and Multimedia Networks, 2018.

- Ascigil, O., Phan, T.K., Tasiopoulos, A.G., Sourlas, V., Psaras, I. and Pavlou, G., 2017, December. **On Uncoordinated Service Placement in Edge-Clouds.** In Cloud Computing Technology and Science (CloudCom), 2017 IEEE International Conference on (pp. 41-48). IEEE.

- Tasiopoulos, A.G., Atarashi, R., Psaras, I. and Pavlou, G., 2017, August. **On the Bitrate Adaptation of Shared Media Experience Services.** In Proceedings of the Workshop on QoE-based Analysis and Management of Data Communication Networks (pp. 25-30). ACM.

- Tasiopoulos, A.G., Kulkarni, S., Arumaithurai, M., Psaras, I., Ramakrishnan, K.K., Fu, X. and Pavlou, G., 2017. **DRENCH: A Semi-Distributed Resource Management Framework for NFV based Service Function Chaining.** In IFIP Networking Conference (IFIP Networking) and Workshops, 2017. IFIP.

- Sha, H., Tasiopoulos, A.G., Psaras, I. and Pavlou, G., 2016, May. **A Collaborative Video Download Application Based on Wi-Fi Direct.** In International Conference on Wired/Wireless Internet Communication (pp. 147-158). Springer, Cham.

- Tasiopoulos, A.G., Psaras, I., Sourlas, V. and Pavlou, G., 2016, May. **Tube streaming: modelling collaborative media streaming in urban railway networks.** In IFIP Networking Conference (IFIP Networking) and Workshops, 2016 (pp. 359-367). IEEE.

- Kulkarni, S.G., Arumaithurai, M., Tasiopoulos, A.G., Psaras, Y., Ramakrishnan, K.K., Fu, X. and Pavlou, G., 2016, April. **Name enhanced SDN framework for service function chaining of elastic network functions.** In Computer Communications Workshops (INFOCOM WORKSHOPS), 2016 IEEE Conference on (pp. 45-46). IEEE.

- Tasiopoulos, A.G., Psaras, I. and Pavlou, G., 2014, September. **Mind the gap: modelling video delivery under expected periods of disconnection.** In Proceedings of the 9th ACM MobiCom workshop on Challenged networks (pp. 13-18). ACM.

Under Submission

- Tasiopoulos, A.G., Ascigil, O., Psaras, I., Toumpis, S., and Pavlou, G., **FogSpot: Spot Pricing for Application Provisioning in Edge/Fog Computing.** Under journal submission.

Chapter 1

Introduction

We are in the midst of a revolution in communication services where a plethora of applications progressively affect different aspects of our daily activities. Recent communication paradigms [33, 80, 193, 206] envision a more immersive and pervasive Internet that will globally enable domains such as home automation, mobile healthcare, elderly assistance, autonomous vehicles, and many others [40, 132]. Over the last decades, the Internet has provided the technological glue for interconnecting people while fostering the conditions for the development of a phenomenal number of applications. The Internet is increasingly a platform for accessing services [139] where different economic network entities, such as Internet Service Providers (ISPs), cloud computing providers etc., interact for delivering services to end users with the support of Network Functions (NFs) that manipulate the network traffic for other purposes than packet forwarding, like load balancing, security *etc.*. However, despite the Internet's commercial triumph in acting as a platform for accessing applications, there is an inherent struggle in providing a satisfying Quality of Service (QoS) [79, 190, 210] due to the "*best effort*" performance of networks' transport services as well as the physical latency caused by accessing remote data-centres, where network applications are hosted. The Internet's structure is bound to inhibit the proliferation of futuristic applications whose QoS relies heavily on underlying latency and network response times. Traditional network and data centre expansion methods present a costly approach [66] that, in the advent of continuing traffic increase, is outpaced in an horizon of few years [96] leading to impractical solutions. Clearly, novel methods with respect to the way different network entities interact to deliver services have to be considered, and *exploiting existing third party computing resources for service provisioning* appears to be a very attractive and incrementally deployable approach.

1.1 Motivation

Service provisioning refers to the allocation of cloud computing resources, in the form of Virtual Machines (VMs) that can host application instances for serving their end users' requests [130]. Service provisioning is enabled via virtualisation, defined as the process of creating an emulation of a hardware and/or software environment that appears to the user as a complete instance of that

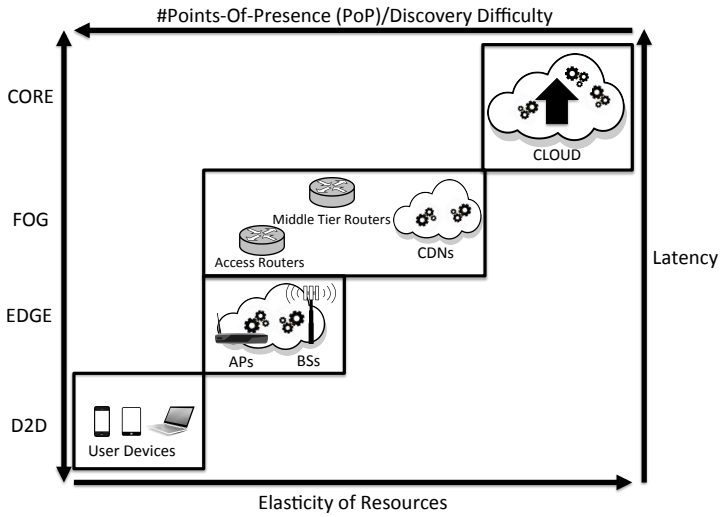


Figure 1.1: In-Network Computing Resources Overview

environment [58]. Virtualisation is a fundamental element of cloud computing [186] where it is used extensively for the manipulation of the involved hardware. In other words, virtualisation enables cloud's elasticity in coping with demand changes, for computing resources, in a cost efficient way via on-demand computation [72]. However, an increasing number of Low Latency Applications (LLAs), characterised by strict network throughput and delay requirements, render the current centralised cloud-based infrastructure unfit for purpose [59].

An application request is typically forwarded to an application instance located at a data centre/cloud leased, or occasionally owned, by the entity in charge of satisfying the end users' requests; we refer to that entity as Application Service Provider (AppSP). The cloud's elasticity guarantees the availability of applications' instances by provisioning additional resources when it is required. However, applications' QoS is still prone to both best effort network performance and the physical latency of accessing the cloud. Especially in the case of LLAs the QoS depends on *i*) the relative position between end users and application instances, in terms of number of hops, latency, congestion *etc.*, due to the correlation between the proximity of service provisioning and the network conditions improvement [48, 144], *i.e.,* the closer a service is provisioned the lower the chances for the best effort transport services to fail, and *ii*) the network functions that operate on the traffic. Specifically, network functions come in the form of dedicated *middlebox* hardware that inspects, filters, and/or manipulates the traffic for purposes other than packet forwarding, related to security and network performance [51]. Nowadays, due to the cloud success in delivering computationally intensive services, middlebox network functions are considered getting virtual [75, 131], referred to as Virtual Network Functions (VNFs), and provisioned in the cloud [172].

We argue that a network service access platform, that aims at improving the experienced net-

work conditions, should ensure that both LLA and VNF instances have numerous options of getting provisioned; improving the underlying best effort network service *indirectly*, by bringing LLA instances *closer* to the end-users, as well as *directly*, by deploying VNFs in a network performance-aware way. That said, a ubiquitous deployment of computing resources is prohibitive due to the associated investment cost, *i.e.,* capital expenditure (CAPEX). A promising alternative that ensures the high availability of hardware infrastructure is the *employment* of third party computing resources, as depicted in Fig. 1.1, that already exist in abundance in the form of *i*) user devices capable of device-to-device communication (D2D), including smart phones, laptops, IoT gadgets, autonomous vehicles, and smart home infrastructures; *ii*) edge computing, existing in (WiFi) access points (APs) and mostly at the base stations (BSs) of the cellular networks [23] as "*data centres in a box*", *i.e.,* cloudlets [163]; *iii*) fog computing [181], covering the computing resources of access and middle-tier locations of the network [181] such as routers and other network equipment, Content Distribution Networks' (CDNs) infrastructures [146], and cloudlets.[1] Clearly, the approach of provisioning services over third party computing resources augments the Internet's capability in delivering LLAs by extending the cloud paradigm to include and dynamically provision any kind of accessible computing infrastructure. Specifically, we refer to any form of hardware infrastructure accessed via the Internet as In-Network Computing Resource (INCR) and to the owners of INCRs as In-Network Computing Providers (INCPs). Research so far has mainly focused on optimising network and transport protocols to reduce latency [81, 100, 110, 140, 204] rather than exploiting the INCRs for the provisioning of applications. *This thesis studies the problem of optimising resource allocation and utilisation of INCR, in order to improve the QoS in delivering LLAs via network-aware service provisioning.*

However, the INCPs are independent entities that need to be incentivised in order to participate, since the LLA provisioning involves operational expenses, *e.g.,* energy consumption. Hence, in order to enable the effective provisioning of services over the INCRs, we have to provide answers to the questions of *how the INCPs are incentivised to participate?*, and/or from an application perspective *how the services manage their instances over the network for satisfying the end users' demand?*

The challenge here is the design of provisioning mechanisms that incentivises the participation of INCPs. The provisioning problem is additionally complex due to the following practical issues that we explicitly take into account:

***A1*) Limited Elasticity of INCRs.** In general, INCRs are incapable of providing the essentially boundless elasticity of the cloud, due to the limited amount of computing resources at their possession; that is, at a given INCR, the demand for resources can exceed their availability.

***A2*) Discovery of INCRs.** Discovering computing resources for provisioning services is a challeng-

[1] The term fog computing is used interchangeably to edge computing in the case of IoT applications in the literature [47]; however, in order to avoid confusion we keep the two terms distinct.

ing task due to *i*) the INCRs' availability, related to INCRs' elasticity limitations, and *ii*) the number of INCRs Points-of-Presence (PoP), which are exceeding by far the corresponding number of Clouds.

***A3*) On-Demand Provisioning.** Applying the paradigm of on-demand computation for application provisioning is not trivial in the case of INCRs due to *i*) the limited elasticity of INCRs, *ii*) the INCRs' discovery challenge, and *iii*) the offered INCRs' QoS gains, since INCRs' locations have an individual impact on the involved QoS of different services.

1.2 Methodology & Contributions

We consider INCPs as independent economic entities who are interested in maximising their (monetary) profit by leasing their computing resources. Specifically, we investigate application provisioning mechanisms that operate in the context of a *market* [11] tailored to the different settings of LLAs and VNFs, where a market acts as a dedicated location to which INCPs can advertise their available resources. The choice of approaching the provisioning problem as a market comes naturally due to the need for a systematic way of discovering available INCRs (*A2*) while incentivising the participation of INCPs. Then, the design of provisioning mechanisms is responsible to handle both the challenges of limited elasticity (*A1*) and on-demand provisioning (*A3*).

In particular, for the provisioning of VNFs, we developed a market based resource management framework for the deployment of network functions in arbitrary network topologies with respect to the virtual middleboxes and network conditions. On the other hand, regarding the LLAs, we consider their provisioning under the scenarios of:

1. *Disrupted Connectivity*, where users have to rely on other collocated users' devices for continuing accessing an application for as long as they are disconnected. Along these lines we investigate the problem of collaborative media streaming as an extreme case of delivering a LLA in disrupted connectivity.

2. *Mobile Connectivity*, where the network conditions with respect to a specific application instance change as soon as users move to a different access point, rendering the current service provisioning outdated. Hence, the service provisioning has to be performed periodically in order to follow the changes of QoS caused by the mobility of users.

3. *Fixed Connectivity*, where despite the stable network conditions between the end users and application instances, we assume that once the users get engaged with application instances they cannot be disrupted. In that way, we study the case of provisioning services that are heavily stateful [45], *i.e.,* a user cannot be redirected to another application instance due to the execution runtime information that has to migrate and would result in a negative impact on her QoS. Therefore, the service provisioning has to be performed with respect to the future demand for LLAs.

Both VNFs and LLAs provisioning takes place with respect to challenges (*A1*)-(*A3*).

The aforementioned connectivity settings cover the spectrum of scenarios under which users access applications. This thesis at first gives emphasis to mobile connectivity by also including the extreme mobile case of disrupted connectivity. In fact, there are good reasons to believe that the majority of next generation applications will be accessed by mobile users, especially if we consider the increasing number of per person wireless devices connected to the Internet in the context of IoT[2]. At the same time, this work would be incomplete without considering the scenario of fixed connectivity. Given that every mechanism applied to the mobile connectivity could be adjusted to the less demanding case of the fixed one, we consider the setting of fixed connectivity under the challenging deployment of heavily stateful applications.

This thesis makes the following contributions:

***B1*) Investigates the problem of Service/Application Provisioning over INCRs.** We study the problem of service provisioning of VNFs, in arbitrary topologies, as well as LLAs, under different connectivity scenarios.

***B2*) Develops Market Oriented Application Provisioning Mechanisms that Incentivise INCPs.** We focus on the interdependent problems of *i*) INCPs monetary profit generation, *i.e.,* how infrastructure owners make money, and *ii*) real-time INCRs management, *i.e.,* which LLA/VNF should be deployed and where, in order to support dynamic application provisioning. Our rational is that satisfied AppSPs will continue deploying their services over the INCRs while the profit generation of INCPs will promote the further expansion of their infrastructure.

***B3*) Augments the Cloud Paradigm by Including INCRs.** We complement the cloud paradigm by enabling the on-demand provisioning of services over the INCRs. In that way, AppSPs of LLAs can deliver their services in a network aware way that improves their involved QoS.

***B4*) Promotes the Internet's Role as a Network Conditions Aware Service Access Platform.** The improvement of QoS produced from the provisioning of LLAs over INCRs stimulates the further development of LLAs. As a result, the Internet enhances its position as a service access platform that welcomes the deployment of LLAs.

1.3 Thesis Outline

The rest of the thesis is organised as follows:

Chapter 2 provides an overview of typical applications' deployment over the network and the challenges of enabling LLAs over INCRs.

Chapter 3 focuses on the collaborative media streaming on urban railways as an extreme case of provisioning LLAs in the context of disrupted connectivity. Specifically, this chapter describes

[2] According to Cisco's Complete Visual Networking Index Forecast.

a feasibility evaluation tool for estimating the performance of a TV/radio channel application where commuters can either "tune in" or submit requests for a content that they desire to watch or listen to, which eventually forms a playlist of videos/podcasts/tunes. Given that connectivity is challenged by the movement of trains and the disconnection that this movement causes, users collaboratively download (through cellular and WiFi connections) and share content, in order to maintain undisrupted playback, having at the same time the joint role of INCPs that host a LLA and the role of end users who request it.

Chapter 4 describes a mechanism tailored to the emerging market of LLA provisioning over the INCRs with respect to the end users *mobility*. Along these lines, we propose an Edge Market mechanism (Edge-MAP) for bringing together INCRs from various network locations and mobile users via cellular based markets. Edge-MAP at a micro-level relies on Vickrey-English-Dutch (VED) auctions to perform robust resource allocation, while at a macro-level fosters the competition among INCRs by providing feedback regarding profit opportunities. The provisioning mechanism is executed periodically to adapt the existing service provisioning to QoS changes caused by end users' movement.

Chapter 5 presents a market-based provisioning mechanism for heavily stateful LLAs and users under fixed connectivity. We introduce FogSpot, a spot price charging mechanism for on-path, on-demand, application provisioning. In FogSpot, cloudlets offer their resources in the form of VMs via collocated, to the cloudlets, markets that interact with forwarded users' LLA requests in real time. FogSpot associates each cloudlet to a spot price based on the current application requests conditions. The proposed mechanism takes into account the particular characteristics of cloudlets resource provisioning, such as limited elasticity of resources, and LLA attributes, like QoS gain and expected user engagement duration. Lastly, FogSpot guarantees end users' requests truthfulness while focusing on maximising either the cloudlet's revenue or resource utilisation.

Chapter 6 addresses the provisioning of VNFs. Specifically, we introduce DRENCH as an algorithmic resource management framework, designed for arbitrary network topologies, *i.e.,* ISPs, that captures the involved trade-offs of VNFs in minimum workload, load balancing, and flow path stretch. DRENCH operates as a low complexity VNF provisioning and flow steering management framework with respect to the network conditions. Each ISP is considered as a regulated market environment where INCPs can participate and compete by managing their resources for instantiating VNFs that operate on the network traffic at a predefined, by the ISP, price.

Chapter 7 summarises the findings of this thesis before identifying future research directions.

Overall, this thesis covers resource allocation and management of INCRs from a multitude of

viewpoints. Starting from delivering stateless LLAs over user devices under disrupted connectivity (Chapter 3), moving to stateful LLAs for mobile users (Chapter 4) followed by heavily stateful for fixed users (Chapter 5), to conclude in the deployment of VNF (Chapter 6). We summarise the contributions of this thesis while pointing out future research directions in Chapter 7.

Chapter 2

Enabling Network Applications: Overview and Challenges

In recent years, the Internet has been increasingly used as a platform for accessing services/applications, such as social media, on-demand video, online games *etc.*The purpose of this thesis is to provide the algorithmic framework details that enable applications that under the current network application deployments mechanism cannot achieve a satisfying QoS. In this chapter, a review of the related work in the literature is presented. At first, we provide a brief overview of network applications' architecture, accessing process, and engagement in section 2.1. In section 2.2, we focus on the provisioning of applications over data centres, as the initial resource allocation step in enabling a network application, followed by a summary of alternative computing resource usage in section 2.3. Then in section 2.4, we present the involved challenges in deploying LLAs over the network. We conclude in section 2.5 by pointing out the contributions of this thesis.

2.1 Network Applications

An application, or application program, is a software designed to perform a group of functions that benefit their user. Applications fall into *network* and *stand-alone* categories. In stand-alone applications, both the functions and the data of the software are stored locally at the user's computer. On the other hand, in network applications software functions and/or data reside in remote computing facilities accessible via a network, *i.e.,* the Internet. In this section, we outline the structure and operation of network applications, or simply applications, by describing at first their client-server architecture before focusing on the user application access and engagement process. Clearly, accessing an application requires the deployment of the application over the network that takes place during the application provisioning phase explained in section 2.2.

2.1.1 Client-Server Architecture

Typically, an application follows a distributed structure that partitions tasks between the providers of a service and the service users, *i.e.,* clients, defining a client-server execution model [159, 167].

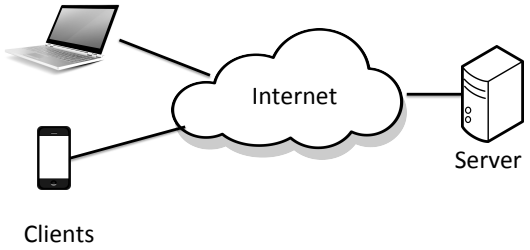


Figure 2.1: Client-Server Model

The client-server design enables the hosting of the service and client functionalities on separate hardware that communicate over a computer network, *i.e.,* the Internet, as depicted in Fig. 2.1. That is, servers await incoming application requests while the client's role is restricted to requesting application functions. The client-server model characterises the collaboration of different application components. In detail, clients and servers exchange messages based on a predefined protocol that dictates both the language and rules of their inter-process communication. Client-server protocols operate at the application layer of Open Systems Interconnection (OSI) stack as depicted in Fig. 2.2, where data is exchanged by traversing a sequence of layers before being transmitted in the form of bits. The logical decoupling of the client-server model comes at the benefits of:

- **Centralisation of control:** All the data related to an application are managed by dedicated servers so that unauthorised clients cannot damage the system.
- **Scalability:** The number of servers can be increased at any time based on the clients' demand.
- **Clients requirements relaxation:** The computing and storage requirements at the client side are reduced since the requested computing functions can be executed at the server's side.

On the other hand, the client-server model involves also disadvantages such as:

- **Traffic Congestion:** The availability of an application's server component is prone to the volume of client requests that attempt to access an application simultaneously. In detail, if the number of client requests exceeds the capacity of the server infrastructure, the application becomes unavailable. The same principle is exploited in the case of Denial-of-Service (DoS) cyber attacks [133] where the perpetrator artificially initiates a sufficient number of sessions that render the application unaccessible to authorised clients.
- **Purchase & Operational Servers' Cost:** The cost of purchasing and operating a server is significantly high since it has to include *i*) the cost of hardware, *ii*) the cost of operating systems and applications, and *iii*) the cost of administer. Therefore, most application providers instead of building their own data centres prefer to lease third party computing infrastructures, *i.e.,* the cloud.

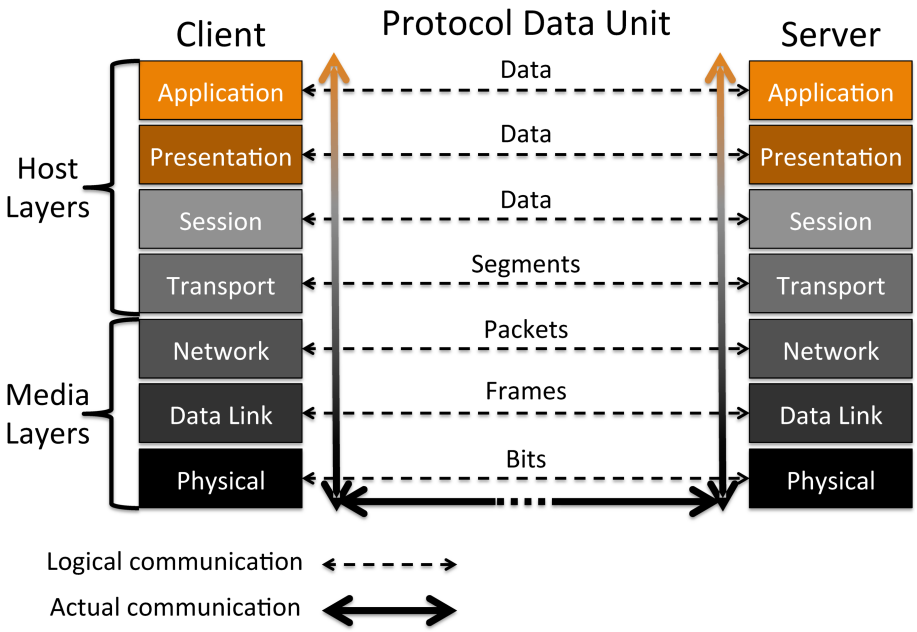


Figure 2.2: OSI stack

- **Network performance:** The inter-process communication delay between the client and server components depends on the conditions of the network and the latency caused by their physical separation. That is, slow inter-process communication has a negative impact on applications' QoS.

Overall, the client-server architecture comes at unique advantages that despite its shortcomings render it as the dominant model for deploying network applications.

2.1.2 Network Applications Accessing and Engagement

According to the client-server model, clients initiate communication sessions with the servers that host the requested application. At first, the client has to identify the location of a server that the target application resides. A common approach for the client is to use an out-of-band lookup mechanism, like the Domain Name Servers (DNS) [136], which returns the address of such a server. Such a process is referred as application resolution. After that, a communication session between the client and the server components is established before initiating the usage of the application, *i.e.,* user-application engagement time. The application utilisation phases from the client perspective are depicted in Fig. 2.3.

Throughout the user engagement to an application, the connection is supported by Network Functions (NFs) which come in the form of network appliances, *i.e.,* middleboxes, that transform, inspect, filter, or generally manipulate the user's traffic for purposes other than packet forwarding [51]. Middleboxes are deployed primarily for *security*, *e.g.,* as applications firewalls, and *network performance*, *e.g.,* Wide Area Network (WAN) optimisers/load balancers, purposes. That is, middleboxes

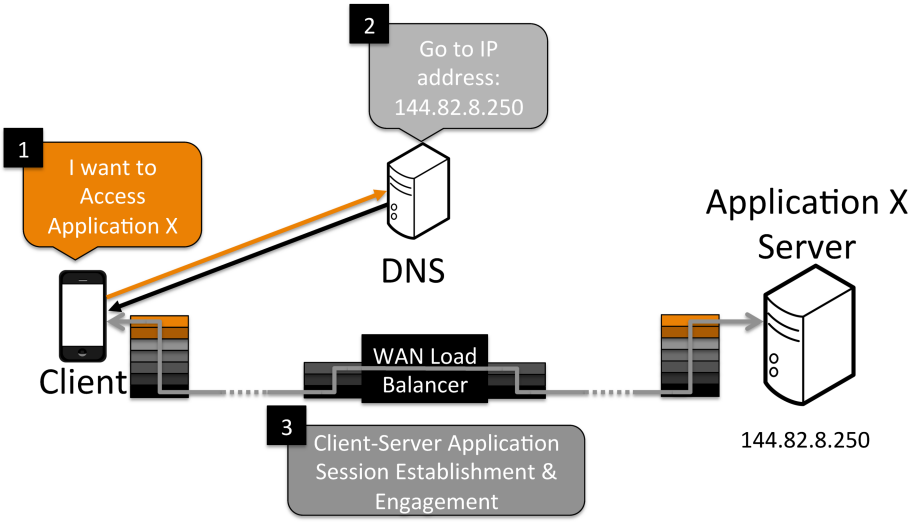


Figure 2.3: Application resolution, session establishment, and engagement example

address partially the weaknesses of the client-server model with respect to traffic congestion, *e.g.,* firewalls can deal with DoS attacks, and network performance, *e.g.,* the load balancers improve the network conditions. Therefore, the QoS of an application strongly relies on the efficient operation of middleboxes.

2.2 Application Provisioning

Application provisioning refers to the set of actions that prepare a server with the appropriate software and data to accept incoming application requests. The provisioning procedure consists of the steps of *i*) server selection from a pool of available servers, *ii*) application related software loading, *e.g.,* drivers, operating system, middleware *etc.*, *iii*) configuration of the system that create/modify the application boot image [149] at this server, *iv*) configuration of the network parameters, *i.e.,* IP address, and storage/computing resources, *i.e.,* resource provisioning, *v*) restarting of the system and loading of the new image. Typically, the boot images are verified by third party organisations or the corresponding application providers and they are capable of creating a virtual instance of an application as soon as they are mapped onto the physical resources of a server.

In order to reduce the dependencies on IT staff members, software products are available that automate the provisioning of applications [65]. Especially when it comes to cloud computing, application provisioning can take place in an automated cost efficient way that enables the self-provisioning of an application.

2.2.1 Cloud Computing Provisioning Benefits

According to the definition of the National Institute of Standards and Technology (NIST) [130]: "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.,* networks, servers, storage, applications, and

services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.". In other words, cloud computing is recognised by NIST as the next generation's computing infrastructure that brings the advantages of:

- **Self-provisioning of resources:** Where a consumer can provision computing and storage resources automatically without requiring human interaction.
- **Broad network access:** Cloud resources are offered over the network to a set of heterogeneous client devices, *e.g.,* mobile phones, tablets, laptops, workstations etc.
- **Resource pooling:** Computing resources are dynamically pooled to serve multiple requests by assigning virtual and physical resources based on the demand.
- **Rapid elasticity:** Resources can be rapidly allocated and released. In that way, resource provisioning can elastically cope with demand fluctuations for resources.
- **Measured service:** Resource usage can be monitored providing transparency to applications providers regarding the utilised servers.

Clearly, cloud computing offers a cost efficient solution to application providers that cannot afford the construction and maintenance of their own data centre.

2.2.2 Cloud Computing Services

Cloud computing is also differentiated from other computing solutions [83] due to its offered allocation guarantees in computing resources, accessibility over the Internet by any device, and the variety of offered services [93], such as:

- *Infrastructure as a Service (IaaS):* IaaS [42] enables the provision of storage, hardware, servers, and in-cloud networking components.
- *Platform as a Service (PaaS):* PaaS offers an advanced integrated environment for building, testing, and deploying customised applications.
- *Software as a Service (SaaS):* SaaS [49] supports a software distribution with specific requirements. In this layer, the users can access an application and information remotely via the Internet. SaaS is sometimes referred to as "on-demand software".

In the case of application provisioning, cloud resources are offered to clients in the context of SaaS. In particular, application server component boot images are loaded to create a virtual instance with dedicated CPU, memory, and storage resources creating a Virtual Machine (VM) mapped onto Physical Machine (PM) resources.

2.2.3 Cloud Computing Pricing

Virtual machines are coming in a sufficient number of different types, in terms of resources, such that their charging mechanism is simplified on per VM type usage for a predefined duration. For example, Amazon [27] offers instances under three pricing schemes:

- Reserved instances, that guarantee the long-term availability of an instance, *i.e.,* more than a year, by charging a fixed usage-based price.
- On-demand instances, that guarantee the short-term availability of an instance, *i.e.,* for an hour, by also charging fixed usage-based prices.
- Spot instances, that create an auction-based market for *spare* instances. In detail, users can use spot instances only if their bids exceed a spot price while spot prices are updated every 5 minutes; meaning that the availability of instances is not guaranteed.

2.2.4 Cloud Application Provisioning Challenges

Application provisioning takes place in the form of requests for VMs, that will host the virtual instances of the corresponding application. Cloud providers accept the requests under their offered pricing schemes and map VMs onto PMs. Initial works on the VM mapping to PM problem assumed that VMs are assigned static shares of PM resources (*e.g.,* CPU, memory). This problem is equivalent to the NP-hard vector bin packing problem [63], which can be used for modelling static resource allocation problems. According to vector bin packing problem the optimal mapping of VMs is the one where the VMs are *packed* into a minimum number of PMs, with respect to PM resource constraints. However, due to NP-hardness, practical VM placement solutions involves heuristics [90, 106, 121, 142, 194]. More recent works of VM mapping include aspects regarding security constraints [101, 173] and VMs resiliency [43, 69]. Other studies include in-Cloud network bandwidth restrictions [44, 101, 103, 175, 211] and energy consumption [68, 143, 191, 200], where cloud providers try to jointly optimise VMs' mapping with network and/or energy resources within their data centre.

Cloud Computing Energy Consumption and VM Migration

Cloud computing resources are organised in a Fat-tree topology, for provably efficient communication [120], as depicted in Fig. 2.4. Servers are organised in racks, each one forwarding its traffic to the top of rack switch. A layer higher, aggregation switches collect the traffic from the racks and forward it to the core switches of the Internet. The data centre sector is estimated to account for 1.4% of the global electricity consumption [156] with 40% of this being associated (on average) to the cooling system of the data centre [114, 138]. Clearly, energy consumption is an important dimension [34] that has to be included explicitly in the management of cloud resources. That is, a cloud has to be capable of deactivating a set of server racks by following a tradeoff between their performance and the involved energy consumption.

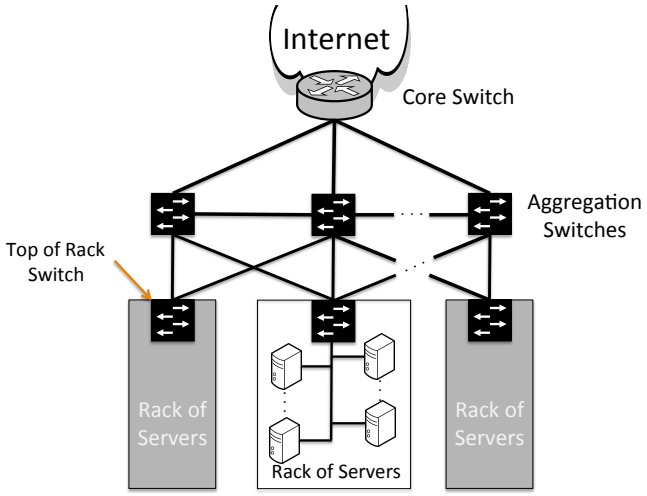


Figure 2.4: Cloud Fat Tree Topology

Live migration [61] refers to the process of moving a running VM to a different PM without disconnecting the client. Live migration is a powerful tool used by cloud providers in order to consolidate servers as well as server racks with the aim of either load balancing the usage of cloud resources [50] and/or reducing their running operational cost [170], *i.e.,* energy consumption. VM migration techniques are categorised into pre-copy and post-copy memory migration.

In the pre-copy memory migration [95], a warm up phase takes place where the memory pages are copied from the source PM to the destination PM while the VM is still running at the source. The warm up phase is followed by a stop-and-copy phase, where the VM stops at the source and consequently it is instantiated at the destination. We refer to the memory pages that changed during the process as "dirty-pages" and the time between stopping and instantiating the VM as "down-time". Down-time varies from milliseconds to seconds given the memory size of the dirty-pages, that have to be copied at the destination before the instantiation of the VM.

On the other hand, in the post-copy memory migration [92], a VM is instantiated at the destination after suspending the VM at the source without transferring any memory pages, *i.e.,* "down-time" is zero. After the completion of the instantiation at the destination, memory pages start getting transferred from the source as a background process. Clients that attempt accessing an unavailable memory page receive a page fault message and they are redirected to the memory of the source. However, the retrieval of the missing page comes at the expense of memory access cost/delay that degrades the performances of the application.

2.3 Provisioning Potentials in Alternative Computing Infrastructures

Edge and fog INCRs consist of routers, network equipment, and existing computing resources deployed over the network, like in the case of Content Distribution Networks (CDNs) [185], which can be exploited for provisioning applications. Additionally, there is a current trend of deploying Commercial-Of-The-Shelf (COTS) computing resources, *i.e.,* cloudlets, closer to the end users. Cloudlets have been proposed in [163] as a surrogate infrastructure where mobile devices can offload computationally intensive tasks to complement their computing capabilities and battery limitations [37, 67]. Hence, research has been focused on task offloading technologies [60] with the aim of augmenting the mobile devices' computing capabilities [161, 205] and/or battery duration [38]. Closer to our setting, [32] addresses the problem of application-specific task offloading over the fog infrastructure, *i.e.,* a task requires the corresponding virtual instance of the application at a cloudlet before getting offloaded. In particular, applications are provisioned according to caching techniques in an uncoordinated fashion for serving tasks on their way to a default execution location. However, the economic aspects of the problems are not considered while the presented caching approaches are suitable for stateless applications which requests can be decomposed to individual tasks that can be executed by any available application instance in the fog.

On the other hand, numerous user mobile devices can be used as a computing infrastructure for delivering applications under the conditions of:

- Lightweight applications, where the computing requirements of an application can be satisfied by the limited capabilities of a mobile device.
- Disrupted (or absence of) connectivity, where it is challenging or impossible for a user to access an application via the network. Clearly, if connectivity is available an application has always a preference of being provided by a computing infrastructure, like a data-centre, where there are no resource limitations, like battery constraints as in the case of mobile devices.
- End users collocation, since the client connects to the server component that is hosted to another device. That is, the distance that a device-to-device session can be established is restricted by the technology used,*e.g.,* bluetooth or WiFi, and therefore both devices have to be collocated.

Exploiting end user devices for sharing data has been studied extensively in the setting of identifying users who possess data of interest to the others. Under the assumption that users stay within the transmission range of the others, they can successfully transfer their content via device-to-device connections as described in early works like BlueTorrent [107]. That said, realistic settings where users remain collocated for a sufficient period of time while experience connectivity disruptions are limited.

Authors in [129] identify public transportation as an environment where commuters can take advantage of other users' devices located in their vicinity. Specifically, based on the concept of *Familiar Strangers* [147], they analyse commuter traces of public transportation systems to investigate whether collocation patterns exist. After verifying their collocation hypothesis they build a device-to-device network for data sharing, regarding the distribution of data stored in the commuters' devices. On the other hand, authors in [155] propose the installation of a hub to the public transport infrastructure for enhancing the connectivity of commuters while [118] focus on the utilisation of such equipment for improving the delivery of streaming applications. Although the existence of a connectivity device on board of a public transportation vehicle, *e.g.,* a train or a bus, is desirable the authors do not exploit aggregate connectivity and storage opportunities offered by user devices.

Device-to-device collaborative downloading has been studied extensively in the context of Delay-Tolerant Networks (DTNs) [77], where some works focus on taking advantage of the social links that drive connectivity decisions [46, 94] while [28, 182] targeted the improvement of downloading speed. Other studies investigate co-operative streaming download techniques for mobile users. Such studies are applicable to a broader scope of problems in the context of delay-/disruption-tolerant networks. For instance, co-operative techniques that exploit cellular and local WiFi connectivity have been studied to improve the capacity available to mobile users, or in other words, increase the aggregated downlink rate of each user [91, 97, 180, 192]. In [177], a group of collocated users, using several wireless Internet access links, jointly access a video stream before sharing their downloaded content with the rest. In [164], the authors formulate a network utility maximisation problem where a single static group of commuters attempts to access a video stream. According to [164], users try to utilise their Internet access links as well as the device-to-device capacity by taking into account the factor of packet loss and applied network coding. Last, but not least, authors in [112] design, implement, and evaluate *Microcast*, a system that improves the streaming experience for a group of users, albeit in relatively static connectivity conditions.

2.4 Low Latency Applications

Cloud is the prominent choice for the provisioning of network applications that follow the client-server architecture. However, applications deployed in that way suffer from *i*) *high latencies*, caused by the physical separation between the client and the server component [171], and *ii*) *unreliable network throughput*, related to the best effort performance of the network. Specifically, both factors of high latencies and throughput fluctuations unavoidably inhibit the proliferation of recently developed applications in IoT (*e.g.,* health monitoring), entertainment (*e.g.,* Virtual/Augmented reality), and automotive (*e.g.,* self-driving cars) domains. We refer to the network critical applications that cannot achieve a satisfying QoS due to dynamic/unpredictable network conditions and the physical latency caused by their users' engagement to the cloud as *Low Latency Applications* (LLAs). In this section, we review current efforts and possible directions that enable the provisioning of LLAs over

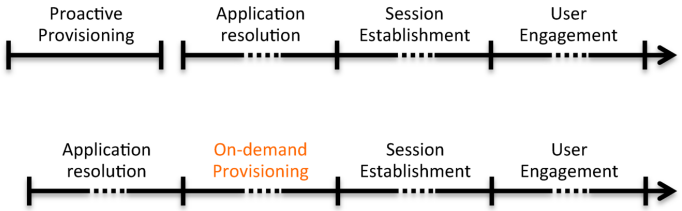


Figure 2.5: Application accessing phases flow, Proactive vs. On-demand provisioning

alternative, to the cloud, computing infrastructures that are physically located closer to the end users.

2.4.1 LLA Provisioning

Provisioning Challenges

INCRs can support a limited number of application instances compared to the cloud. Therefore, the aim of LLA provisioning is to support the LLA instances that achieve the highest possible QoS gain given the upcoming user application session requests. However, the typical proactive provisioning is incapable of achieving the highest possible utilisation of the resources for a given INCR. For example, if an INCR can support only a single instance and currently has proactively provisioned application *A*, it will fail to serve a request for application *B* even if there is currently no one engaged to application *A*'s instance. Hence, the provisioning of LLAs has to ideally take place on-demand, *i.e.,* application B is instantiated as soon as the application resolution directs a request to a computing infrastructure, as opposed to proactive provisioning that takes place before the arrival of an application's request to the infrastructure as depicted in Fig. 2.5. Thankfully, recent developments in lightweight, compact, single address space, memory-safe virtual machines (VMs) written in a high-level language, *i.e.,* unikernels, enable the on-demand instantiation/provisioning of an application [126]. Specifically, in [125] the authors restrict the on-demand applications' provisioning overhead to 350 ms, by masking the image booting latency, even for the most demanding boot images.

Hence, the main challenge in on-demand provisioning is related to the management of on going application sessions. For example, if the upcoming request of user 1 achieves a higher QoS gain than the currently engaged user 2 at a given INCR, the dilemma is whether user 2 should be disrupted for serving user 1. Clearly, in the context of LLAs suspending the engagement of a user to an application is not an option, which means that in order to disrupt a user, we have to ensure the live migration of her instance to a different INCR. That is, the migration between different INCRs is associated to transferring VM memory pages over the Internet, instead of cloud racks. Therefore, going back to the initial question of disrupting a user for serving another, the answer depends on the memory pages of an application. In particular, we have the following cases:

- Stateless LLA, where there are no memory pages to be transferred: the users can be disrupted at any time assuming there is always a backup VM that can serve them.

- Acceptably Stateful LLA, where the size of memory pages can be transferred to another location in a reasonable amount of time: the mapping between users and INCRs could be re-estimated periodically.

- Heavily Stateful LLA, where the size of memory pages prohibits the migration of a user to another location: the users cannot be disrupted and INCRs have to be rigorous with the requests that are accepted to be served in the first place.

The situation becomes even more challenging in the setting where mobility is considered. In the case of mobile users, network access points evolve following the movement of the users, rendering the connection to a given INCR outdated since the initial latency as well as network conditions change over time, which has an impact on the LLAs' QoS.

That said, the most important aspect of on-demand provisioning is to derive the appropriate mechanisms for incentivising the participation of computing resources. In general, for INCRs a pricing mechanism is crucial for quantifying the monetary income they can obtain by participating. The providers of INCRs consist of diverse third party entities that we assume they are having monetary incentives in leasing their resources for the provisioning of LLAs. Therefore, both the providers of LLAs and INCRs have to be capable of estimating the usage make of each INCR by each LLAs' users by applying commonly acceptable pricing schemes. Thus, every on-demand provisioning mechanism has to take place in an economic context that is expressed in the form of resource pricing.

Pricing of Provisioning in Alternative Computing Infrastructures

Despite the rich literature regarding the pricing aspects of cloud resources, such as [25, 169, 199], there are limited efforts covering this topic in the case of INCRs especially in the context of on-demand provisioning. Specifically, the authors in [104] present a centralised double auction scenario for offloading a fixed set of tasks over a set of cloudlets, without considering however the problem of application provisioning. Closer to LLA provisioning, in [117] authors propose a self-tuning service/application provisioning combinatorial auction mechanism, where application providers bid periodically for a specific number of VMs organised in different execution zones of identical INCRs. However, the presented mechanism relies on precise predictions about the future demand of an application while the optimal bidding derivation is a computationally expensive process that questions the scalability of such a scheme when it is applied in short time intervals. Furthermore, their design is based on the proactive provisioning of resources, leaving the resolution of each request to the provider of each application.

2.4.2 LLA Resolution

Resolution Challenges

Today's network stack overloads the meaning of addresses and port numbers in order to differentiate connection end-points and apply the requested application protocols. In this section, we explain the limitations of the network stack in supporting applications that operate under multiplicity, in application instances, and dynamism, due to application instantiation/consolidation in different locations of the network, and client mobility.

In particular, the *application layer* relies on IP addresses and TCP/UDP ports to implicitly refer to applications/services. That is, clients use an out-of-band lookup mechanism, like DNS, combined with a priori knowledge, *i.e.,* HTTP service is running on port 80, in order to establish a session with an application instance. Clearly, AppSPs have to register any changes regarding their available application instances into the out-of-band lookup mechanism in order to be accessible to the users. The scalability of the resolution process is questioned when it comes to LLAs since the extended multiplicity and dynamism of LLA instances is expected to degrade significantly the performance of such an approach.

Furthermore, since the *transport layer* relies on a five-tuple (remote IP, remote port, local IP, local port, protocol) to demultiplex incoming packets to a socket, it prohibits any modification onto the interface addresses without suspending the client-server session. Therefore, the currently deployed stack is incapable of supporting mobile users without disrupting their ongoing connections upon changes of their IP address caused by their movement to another access point.

Service Centric Networking Resolution Approach

The main disadvantages of the resolution mechanism stem from the overhead caused by the coupling of an application ID to its host. Clearly, this is a deficiency of the network stack's design that has been identified and studied in the context of Information Centric Networking (ICN) [4].

ICN has been proposed as a candidate for the Future Internet architecture [22, 57, 201] that essentially decouples information from its hosts by naming the information explicitly, turning the resolution operation into a *receiver-driven* process where users request information, or services, explicitly by their name. The idea of ICN design was introduced in the context of content retrieval in the seminal papers of [89] and [53], as well as in other early works such as [52, 54]. ICN potentials include:

- *Information delivery merits:* Since an in-network content aware mechanism can be deployed in order to identify the information location which offers the most promising QoS.

- *Users mobility support:* Users can retrieve information as they move without experiencing connectivity disruptions upon their handover to a new access point. The reason is that each chunk of content is retrieved by name instead of relying on the five-tuple of transport layer.

- *Security features:* In ICN malicious or spamming data filtering can take place via self-certifying information naming. In addition, DoS attacks cannot target specific hosts anymore and therefore it is much harder for them to render an information unavailable.

Numerous projects have illustrated part of ICN architecture merits [1, 4, 5, 7, 8, 9, 10, 12, 208, 116]. That said, only COMET [6] includes aspects of ICN advantages in the improvement of the content's delivery QoS. In particular, COMET defines 3 classes of services (CoS) that prioritise the end-to-end communication traffic. Clearly, this approach improves the QoS of a connection given the placement of the content. Open issues in ICN concern the *i*) content deletion; *ii*) versioning incorporation, so multiple versions of the same content can be available; and *iii*) name scalability issues, since each chunk of each content requires a unique ID.

However, here we are interested in the benefits of ICN principles for decoupling application identifiers from their hosts in order to improve the accessibility of LLA instances. Service Centric Networking (SCN), as defined in [84], refers to the name based applications resolution. The application name space of this setting is much smaller than the one required for every chunk of every streaming content, *i.e.,* only a single name ID for each application version would be sufficient. Therefore, SCN enjoys the merits of ICN without the disadvantages that are caused by the massive namespace of each content.

In more detail, authors in [166] present Services-over-Content-Centric Routing (SoCCeR) approach as a straightforward extension of CCN [4] with integrated support for service routing. On the other hand, a more incrementally deployable approach is demonstrated in [139] where the authors revisit the design of the network stack in order to allow applications to communicate directly by their name. They introduce Serval, a Service Access Layer that is deployed on top of an unmodified network layer while being capable of providing a service-level resolution via diverse discovery techniques. In Serval, end-points can seamlessly change network addresses while establishing additional connections that assure the efficient and undisrupted service access. Lastly, a Keyword-Based Mobile Application Sharing (KEBAPP) is presented in [151] as a SCN approach where users can share the applications hosted on their mobile devices with nearby users. Keyword-based requests are becoming useful for discovering applications in crowded areas and/or in the advent of disrupted network connectivity.

2.4.3 LLA User Engagement

Engagement Limitations

As mentioned before, both network operators and users require various additional functionalities in the network for the management and processing of data flows. These functionalities are provided in the form of middleboxes [51, 188] with the purpose of policy control, security, and network performance optimisation. Middleboxes are typically developed as purpose-built hardware, customised to perform specific tasks. Once setup, a network of middleboxes cannot alter its structure (*e.g.,* topol-

ogy) or (service) functionality (*e.g.,* morph from one service to another). One of the big challenges in network management is that of *Service Function Chaining* (SFC) where flows are steered through different middleboxes before being delivered to their final destination. Traditional solutions for SFC are based on redirection and policy routing. In particular, in approaches like [36, 188], we see architectural modification to TCP/IP networks capable of allowing middlebox redirections, while other solutions [178, 187] rely on predetermined middleboxes that however remain incapable of reacting to middlebox failures.

LLAs like the rest of network applications potentially rely on network functions to secure their smooth operation throughout the phase of their user engagement. SFC does however negatively affect LLAs since the imposed flow steering deviates a connection from its shortest path with the outcome of increasing the latency between the client and the server components. That is, middleboxes ideally should be deployed at network locations that do not impose any latency and bandwidth overheads on LLA users' engagement. However, LLAs' multiplicity and dynamism is challenged by the stationary deployment of middleboxes.

Support User Engagement via Network Function Virtualisation

A realistic approach for dealing with this challenge is to complement users' engagement to LLAs by provisioning virtual versions of NFs over the computing resources that host LLAs. Specifically, Network Function Virtualisation (NFV) [56] has been proposed to increase flexibility in the network, evolving middlebox architectures to virtual, as software-based services on top of COTS hardware, which in our case could be any kind of INCR. NFV promises to improve both the efficiency of computing and network resources by dynamically altering the structure and functionalities of NFs in response to their demand. That said, the flexibility of adjusting the number of instances for each NF comes at the complexity of SFC since steering decisions have to be updated upon the instantiation/consolidation of a NF. Specifically, typical approaches for SFC require manual configuration on specific routers for applying adjunctive rules. Clearly, this approach does not scale well in the dynamic conditions of NFV. Only in [31] the authors suggests a *Function Centric Service Chain* approach that decouples the functions a flow needs from their location (and thus do not assume fixed routing paths) via a naming layer based on the principles of ICN. In that way, they propose the naming layer for building on top of that flexible as well as scalable solution for SFC in the dynamic environment of NFV. Other works, like FlowTag [78], SIMPLE [152], Stratos [87], PLayer [105] are also examples of approaches that provide control and coordination of traffic forwarding.

In terms of virtual network functions instantiation/consolidation, a significant number of works consider NFV nodes load-balancing of computing resources in data-centres applying centralised management approaches, like Hedera [24], MicroTE [41] and F10[123]. Other works, such as Split/Merge [154] and Pico Replication [153], are designed to provide load-balancing as well as flow redirection. The downside is that there exists a race condition between updating network state

and resuming flows (flows are halted when states are changed for active flows) resulting in loss of packets. OpenNF [88] enhances Split/Merge and Pico to overcome the race condition but still the complexity of taking load balancing and redirection decisions is prohibitive for real time decisions.

Regarding distributed NFV management approaches, Ananta [145] is a software that operates on the data plane of data-centres for splitting the incoming traffic to different NFV instances. Duet [86] enhances Ananta by using a mixture of software and switch-based load balancer while NetAgg [127] is a similar distributed approach designed to aggregate application specific data. CONGA [26] proposed a distributed load-balancing technique between ingress-egress routers based on path specific congestion metrics. That is, CONGA operates at microsecond time-scales since it can be implemented in switch hardware.

2.5 Summary of LLA Enabling Aspects Covered in this Thesis

The majority of this thesis' contributions are coming in the form of economic-based algorithmic frameworks that incentivise the participation of INCRs for provisioning LLAs with respect to the INCRs limitations in different connectivity scenarios. LLAs are expected to be delivered like any other network application that follows a client-server architecture. The main challenge here is that the typical application provisioning into distant Data Centres/Cloud fails to provide the required response times by the LLAs, with the result of offering a degraded quality of delivery. The idea of exploiting existing INCRs for provisioning LLAs, as an alternative computing infrastructure to the cloud, seems as a promising research direction. However, provisioning LLAs over INCRs challenges the standard resolution process, a problem that we will not address explicitly in this thesis. Instead, we will consider other efforts, regarding a scalable and adaptive resolution mechanism, orthogonal to our work while identifying ICN approaches, that address LLA instances by name, as the most feasible one. Other difficulties are related to the operation of middleboxes on LLA sessions, since middleboxes tend to deviate flows from their shortest paths, negating the benefits of LLA provisioning over the INCRs. We overcome this difficulty by relying on NFV for managing virtual network functionalities over INCRs. In that way, VNF are capable of following the dynamicity of LLA instances with respect to their impact on the LLA connections response time requirements.

Incentivising LLA On-Demand Provisioning

We consider incrementally stateful LLA requirements deployed under different user connectivity conditions.

We start from stateless LLAs for which delivery is of particular interest when it takes place via other users' mobile devices. As motivated in Section 2.3, we focus on the challenging scenario of collaborative streaming in the public transportation setting of disrupted connectivity. In particular, we present a QoS assessment framework that quantifies the QoS gain of collaboration that eventually acts as a direct incentive for the commuters' participation. This work is the first of its kind, as

presented in Chapter 3.

Then, in Chapter 4, we move to reasonably stateful LLAs, *i.e.,* LLA VMs can be migrated to different INCRs in the network, and their delivery to mobile users. We propose a design for a market mechanism tailored to the mobile user demand dynamics that benefits both LLAs' QoS and INCPs' income. We introduce Edge-MAP, as the underlying framework that enables the in-network, on-demand provisioning market for LLAs with respect to the mobility of the users.

Lastly, in Chapter 5 we consider enabling heavily stateful LLAs under fixed connectivity. We introduce FogSpot pricing mechanism, which derives an instantaneous (spot) price for each INCR. FogSpot takes into account both requests' QoS gain and the expected user engagement duration, since a user connection cannot be suspended or migrated, while maximising INCRs' revenue or utilisation. Note that Edge-MAP mechanism can be applied for stateless and stateful LLAs for both fixed and mobile connectivity conditions.

LLA Engagement Contributions

Overall, distributed approaches are not designed to handle a complex SFC scenario wherein a flow has to traverse one or more NFV instances that are dynamically placed in the network, load-balance among active instances, and deal with the efficient placement of VNFs. To the best of our knowledge, despite the work presented in Chapter 6, there are no efforts concerning the joint problem of NFV management and SFC with respect to the latency and throughput aspects of a flow in arbitrary topologies that eventually can enable the support of LLA engagement sessions.

LLA Resolution

This thesis considers the research conducted in the context of SCN, and/or any other approaches addressing the shortcomings of DNS resolution, orthogonal. In the next chapters, we assume the deployment of a underlying SCN approach that is capable of identifying a common streaming channel of interest for collaborative downloading, Chapter 3, or mapping application requests to INCRs for the purpose of on-demand provisioning, in Chapters 4 and 5, or steering flows in the setting of SFC in NFV, Chapter 6, always by decoupling the application/service name from the physical host.

Chapter 3

Enabling LLAs in Disrupted Connectivity Settings

3.1 Introduction

In urban settings the quality of Internet services deteriorates when several hundred users attempt to connect simultaneously through the same WiFi Access Point (AP) [19] or cellular Base Station (BS) [137, 195]. This situation is rather common in metropolitan areas where hundreds of users commute (*e.g.,* trains/buses), wait in stations, or just move slowly towards their destination [124]. The case becomes more challenging when connectivity is physically disrupted, *e.g.,* when (underground) trains travel between stations. In those scenarios, installing more bandwidth will not necessarily improve performance [122], simply because quality often suffers due to: *i)* frequent handovers, *ii)* the hundreds of sessions that an AP/BS has to handle simultaneously, and *iii)* physical challenges, such as long disconnection periods. In this chapter, we investigate the problem of *LLA delivery in environments where end users experience periods of disconnection*.

We argue that in mixed cellular and WiFi access environments the delivery of services can be improved significantly when it is complemented by local Device-to-Device (D2D) transmissions. That said, such a setting requires:

1) The end users' collocation during the period of disconnection. D2D transmissions' distance varies from few meters, in the case of Bluetooth [2], to a few hundred meters, in the case of WiFi-Direct [16]. Hence, an end user has to be located in the proximity of other devices' transmission range in order to access an application.

2) The provisioning of applications that operate under dynamism. Since other devices might arrive/depart dynamically to/from the proximity of a user, the provisioned applications have to be *i*) tolerant to instance failures, since a instance fails when a device departs, *ii*) stateless, since it is crucial for a recently arrived device to provision an instance effortlessly.

In this chapter, we study the problem of application delivery in disrupted connectivity settings

by focusing on the challenging case of collaborative streaming in urban railway networks. Specifically, in urban railway networks, during rush hour, it is quite common for each train to carry more than 500 commuters onboard. Despite the fact that these commuters might want to access a network application, this is quite challenging since connectivity is available for as little as 20 seconds when trains stay in stations, followed by disconnection periods of 4 minutes (or more), when the train is in the tunnel. Urban railway networks offer an environment where connectivity is challenged, by the disconnections that the trains' movement causes, while end users remain collocated throughout the duration of a disconnection, at least until they reach the next station. Furthermore, we focus specifically on the case of collaborative media streaming as a more demanding scenario of stateless application delivery, compared to applications that do not require the continuous engagement of an end user. Thus, we consider end users that have the capability of collaboratively downloading (through cellular and WiFi connections) and share content, in order to maintain undisrupted playback. Clearly, in the absence of any collaboration between users in the process of downloading and streaming content collectively, the users end up with degraded media quality and, more often than not, abandon their sessions.

In detail, we envision a ubiquitous streaming of popular channels service, *e.g.,* national broadcasters, in smart city environments. We assume that commuters either "tune in" to some TV/radio channel, or submit requests for content they desire to watch or listen to, which eventually forms a playlist of videos/podcasts/tunes. The implementation details of such a crowdsourced mobile video and audio on demand streaming service have been studied in [112]. Crowdsourced content retrieval is based on the premise that users share their storage, connectivity and energy resources for performing a common task [73]. In other words, in the context of crowdsourcing media downloading end users act as both INCPs and potential content consumers. That is, the end users participation requires:

A) Direct incentives. In the form of service quality improvement, if the end user is interested in a streaming channel service.

B) Indirect incentives. In the form of monetary reward [180, 197], if the end user can offer her resources for assisting the collaborative content retrieval without being interested in consuming content herself.

In order to enable the effective crowdsourced content retrieval, one needs to answer the following questions: *What is the expected service quality after joining a group that retrieves collaboratively a media streaming channel?* in the case of direct incentives, or *What is the quality gain an end user contributes into a crowdsourced streaming channel?* in the case of indirect incentives. In detail, when monetary incentives take place, the amount of compensation should be determined based on the user's contribution to the media delivery quality gain for the group [1], *i.e.,* the importance of

[1] Like in the case of Shapley Values [158].

the content that she shared with her peers. In this chapter, we fill this gap by building a *quality assessment framework* that answers the above question.

Quality of Experience (QoE) is traditionally characterised by the subjective perception of users [184] and as such, it is difficult to quantify in terms of objective metrics [35]. Indeed, objective measures of user engagement, such as bitrate, throughput, startup delay and buffering, which by and large comprise the state of the art to date, are not adequate in a media-dominated Internet, let alone a mobile environment. The presented framework, which is based on a utility function, goes beyond the above primitive metrics to take into account the user's tolerance to disruptions, the energy needed to download and share media content, as well as the cost of using the cellular network which are crucial factors in the crowdsourcing approach we consider. The energy factor, for instance, can influence the user engagement in collaborative streaming in fear of battery depletion. In other words, users are not expected to spend energy (and/or cellular data) to participate in a collaborative streaming system unless the application quality improvement compensates them. In other words, the introduced utility model is unique in mapping objectives measures onto a subjective quality metric of streaming delivery, that captures the conditions under which a user quits consuming content. As such, we argue that a quality assessment framework for media delivery is necessary to characterise the perceptual and subjective quality as opposed to the network-dependent, objective quality.

The contributions of this chapter include:

- The definition of a utility function as the key aspect of the presented quality assessment framework. This utility function takes into account the playback disruption and energy factors while providing a quantitative measure of a media delivery system for railway networks.

- The analysis of a 17-day sample of commuters' journey traces, provided by Transport for London (TfL, `http://www.tfl.gov.uk`). We identify and illustrate commuting patterns and approximate the number of users travelling in each train throughout the day, as well as the connection and disconnection periods they experience.

- The quantification of the users' experience by applying the utility function to the commuters' traces. Interestingly, we find that although it is difficult to achieve undisrupted multimedia playback, especially during rush hour, a collaborative media streaming application can provide acceptable QoE.

Note that instead of focusing on the development of a monetary reward mechanism for end users that contribute their resources, we investigate the quantification of a user's contribution into the media streaming quality, which works as the basis of her direct or indirect compensation by the channels' AppSP. In other words, we argue that in order to enable LLAs under disrupted connectivity, at first AppSPs have to quantify the improvement in application's utility caused by the participation of other end users in LLAs' delivery. Specifically in the scenario of collaborative streaming, we consider an

AppSP by applying such a framework as a *feasibility evaluation tool* are capable of answering to important questions like *What is the quality achieved for a specific number of channels, bitrate, and energy restrictions for this train?*, and try to optimise these parameters for improving their channels delivery.

The rest of the Chapter is organised as follows. Section 3.2 includes the analytical description of the problem under consideration, while in Section 3.3 we give details of our commuter data trace and we evaluate the performance of the system. Finally, Section 3.4 concludes this study.

3.2 Quality Assessment Framework

3.2.1 System Model

We consider that users access the Internet through either WiFi or cellular links having the options of either individually *Pull* or collectively *PUll and SHare* (PUSH) content. In this section, we lay out the description of the model for each of the Internet access methods considered (*i.e.,* cellular and WiFi), as well as for each content retrieval approach (*i.e., Pull* or *PUSH*). We define as an epoch i, Ep^i, a time interval of duration $|Ep^i|$, which consists of a connection period C^i (of duration $|C^i|$) and a disconnection, or poor connection quality period $\widetilde{C}^i$ (of duration $|\widetilde{C}^i|$). Clearly, $|Ep^i|=|C^i|+|\widetilde{C}^i|$, which also implies that a new epoch starts at each station, where the previous one finishes. Connection period is the time that a train spends in a station and disconnection period is the time it takes for the train to arrive at the next station.

In the simple case, users *Pull* content individually from the Internet. Video or audio content is split in chunks, where every chunk contains y seconds of playback time at bit-rate b. Users can also form groups to *PUll and SHare (PUSH)* content with fellow-commuters in the vicinity. Initially, we present the basic framework which effectively quantifies the utility obtained by individual users (Sections 3.2.2, 3.2.3). We then extend the utility function to incorporate aspects such as energy consumption and cellular download data charges (Section 3.2.4). Finally, we adjust the basic framework to each of the content retrieval approaches (*i.e., Pull* and *PUSH*) and access method technologies in Sections 3.2.5, 3.2.6, 3.2.7.

We assume that the functionality of the $PUSH$ approach, is realised in the context of a mobile Backend as a Service (mBaaS) platform that runs in the cloud (similar in rationale to [112] and [180]) and is responsible for managing group synchronisation in terms of collaborative content retrieval. For the purposes of this study, we ignore any potential implementation overhead (*e.g.,* chunk download scheduling). In particular, the mBaaS platform assigns to each member the content chunks that (s)he has to download and share with the rest of the group participants. All members of a group download and share equal number of chunks which implies *fairness* in terms of computation and communication effort. Our model notation is given on Table 5.1.

$\|C^i\|$, $\|\widetilde{C}^i\|$	i-th connection/disconnection duration.
$\|Ep^i\|$	i-th epoch duration.
A	Internet access technology, *i.e.,* WiFi or cellular.
R	Content retrieval approach, *i.e., Pull*,*PUSH*, or *Hybrid*).
f	Epoch user started downloading the content.
y, b, S	Playback time, bit-rate, and size of a chunk.
Y	Content playback time duration.
$X^i_{A,R}$	Chunks received at epoch i for A and R.
$X^{f\to i}_{A,R}$	Total chunks received until epoch i (inclusive) for A and R.
$L^{f\to i}_{A,R}$	Total playback time received until epoch i (incl.) for A and R.
$W^{f\to i}_{A,R}$	Playback time watched until epoch i (incl.) for A and R.
$D^{f\to i}_{A,R}$	Playback Disruption until epoch i (incl.) for A and R.
$U^{f\to i}_{A,R}$	User utility until epoch i (incl.) for A and R.
$\widetilde{U}^{f\to i}_{A,R}$	User extended utility until epoch i (incl.). for A and R
$U^{C,f\to i}_{A,R}$, $U^{E,f\to i}_{A,R}$	User cellular and energy cost to epoch i.
a_d,a_c,a_e	User disruption, cell, and energy sensitivity.
$Q^{f\to i}_{A,R}$	Efficiency of access technology A and content retrieval approach R.
T_d	Initial Tolerance Interval.
X_{cell},X_{WiFi}, X^g_{p2p}	Per second chunks delivery rate at cellular, WiFi, and group g sharing interface.
$N(t)$	# of users requesting access at moment t.
B_{WiFi}	Total bandwidth assigned to a platform of a station.
B_{Cell}, $\dot{B}_{Cell}$	User good and poor cell bandwidth.
$N_g(t)$	# of users of group g at moment t.
B_{p2p}	Bandwidth limit for local $p2p$ transfers.
$V^{i,g}_{max(p2p)}$	Maximum amount of content to be shared and downloaded over an epoch i for group g.
$V^{i,g}_A$, $\widetilde{V}^{i,g}_A$	Identical chunks downloaded by all the members of group g during good and poor connectivity.

Table 3.1: Quality Assessment Model Notation

3.2.2 Playback and Playback Disruption

The Internet access medium (*i.e.,* WiFi or cellular), as well as the content retrieval approach (*i.e., Pull or PUSH*) affects the number of chunks that a user can receive over a predefined period of time. We denote as $X^i_{A,R}$ the number of chunks that a user receives over epoch i when (s)he accesses the Internet by technology A and retrieves a content by approach R.

Given reception of $X^i_{A,R}$ chunks, we calculate the *playback* and *playback disruption* periods that a user experiences over consecutive epochs. Firstly, we estimate the total number of chunks received by a user over epochs f to i as $X^{f\to i}_{A,R} = \sum_{j=f}^{i} X^j_{A,R}$, where f is the epoch during which the user enters the system.

Next, we express the number of chunks in terms of playback time. We assume that a chunk can be watched/listened only when it has been fully downloaded. Hence, the watching time worth of downloaded content from epoch f to epoch i for one user is:

$$L^{f\to i}_{A,R} = \lfloor X^{f\to i}_{A,R} \rfloor \times y, \tag{3.1}$$

where y is the playback duration of a content chunk.

Given that we model an on-demand streaming service, the downloaded playback time will differ

from the actually watched playback time over epochs f to i, $W_{A,R}^{f\to i}$, due to buffered content. We model this difference in a retrospective manner between epochs i and $i-1$, and express the actually watched playback time $W_{A,R}^{f\to i}$ as the sum of: *i)* the watched playback time during the previous epochs $W_{A,R}^{f\to i-1}$ and *ii)* the difference between the total downloaded playback time between f and i, $L_{A,R}^{f\to i}$, and the actually watched playback time during the current epoch ($L_{A,R}^{f\to i} - W_{A,R}^{f\to i-1}$). Note that in case the user has buffered enough content to get through the current epoch, then $L_{A,R}^{f\to i} - W^{f\to i-1} = |Ep^i|$. Therefore, we have:

$$W_{A,R}^{f\to i} = \min[W_{A,R}^{f\to i-1} + \min(|Ep^i|, L_{A,R}^{f\to i} - W_{A,R}^{f\to i-1}), Y], \tag{3.2}$$

where Y is the total playback duration from the beginning of the journey, which apparently works as an upper bound of the watched time and $W_{A,R}^{f\to i-1} = 0$ for the first epoch.

Finally, the playback disruption time until epoch i, $D_{A,R}^{f\to i}$, is calculated as:

$$D_{A,R}^{f\to i} = \begin{cases} D_{A,R}^{f\to i-1}, & \text{if } W_{A,R}^{f\to i} = Y \\ \sum_{j=f}^{i} |Ep^j| - W_{A,R}^{f\to i}, & \text{otherwise,} \end{cases} \tag{3.3}$$

where $D_{A,R}^{f\to i-1} = 0$ for the first epoch.

3.2.3 Utility Function

The playback time, $W_{A,R}^{f\to i}$, as well as the playback disruption, $D_{A,R}^{f\to i}$, are the fundamental components that we use in order to describe the user's utility in terms of QoE. Clearly, *the ideal utility of a user until epoch i is equal to the sum of the epochs' duration, or the content duration Y*, whichever is shorter:

$$U_{ideal}^{f\to i} = min(Y, \sum_{j=f}^{i} |Ep^j|). \tag{3.4}$$

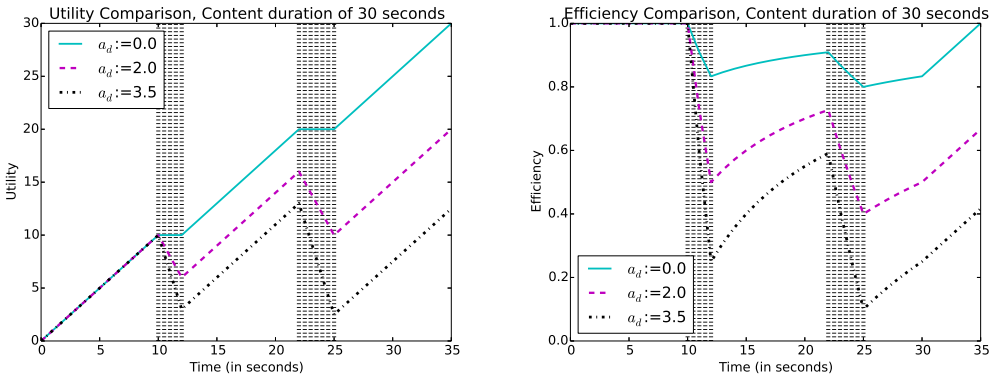
We consider that the utility decreases due to the playback disruption $D_{A,R}^{f\to i}$ and express a *user's utility* in terms of undisrupted playback time according to the formula:

$$Utility\ Function:\ U_{A,R}^{f\to i} = W_{A,R}^{f\to i} - a_d \times D_{A,R}^{f\to i}, \tag{3.5}$$

where a_d is the user's tolerance to playback disruptions. The disruption tolerance factor decreases the user's utility by a_d. For instance, for a user who has watched 10 secs of undisrupted playback and therefore has built a utility function equal to 10, a disruption tolerance factor equal to 2 will decrease his utility to 8, after 1 second of disruption. We consider the disruption tolerance factor as a central component of the utility function for media delivery in connectivity-challenged mobile environments.

We express the "*efficiency*" of an Internet access technology A and content retrieval approach R for a commuter according to the following *utility ratio*:

$$Utility\ Ratio\ /\ Efficiency:\ Q_{A,R}^{f\to i} = \frac{T_d + U_{A,R}^{f\to i}}{T_d + U_{ideal}^{f\to i}}, \tag{3.6}$$



(a) Utility Comparison

(b) Efficiency Comparison

Figure 3.1: Comparison of user utilities and efficiencies over time for a disruption of 2 and 3 seconds after 10 and 20 seconds of playback time.

which is bounded by the interval $[0,1]$. T_d is the "*Initial Tolerance Interval*", which indicates the user's patience to start-up delay. In the rest of this work, we use Eq. 3.6 to quantify the QoE that users obtain during their journeys, which together with the *utility function* in Eq. 3.5 comprise the two main building blocks of the proposed quality assessment framework.

Fig. 3.1 illustrates the *utility* (Eq. 3.5) and *efficiency* (Eq. 3.6) fluctuation for 3 users with disruption tolerance (*i.e.,* a_d) 0, 2, and 3.5, respectively; the users experience a playback disruption of 2 seconds after 10 seconds of playback time and another disruption of 3 seconds after playback time of 20 seconds; the total playback duration is 30 seconds, which also means that the ideal utility is equal to 30. Apparently, a delay sensitivity equal to 0 leads to efficiency equal to 1 (Fig. 3.1b), while for delay sensitivity equal to 2 and 3.5 in this setting the produced efficiency is 0.666 and 0.416, respectively. This result demonstrates that *efficiency* is subjective when it comes to QoE, since it is related to the users' temporal utility/satisfaction and their personal tolerance to disruptions.

The *Utility Ratio or Efficiency* can get negative values due to extended disruptions. In this case, we assume that a rational user would quit attempting to watch (listen) this video (music playlist), which would set the Efficiency value to zero, or more formally:

$$Q_{A,R}^{f\to i,t} = max(\frac{T_d + U_{A,R}^{f\to i,t}}{T_d + U_{ideal}^{f\to i,t}}, 0). \tag{3.7}$$

3.2.4 Cellular and Energy Cost

To further extend our model with a realistic representation of a working system, we incorporate two more factors in the users' utility function. The first is related to the medium used to download content and is associated with the cost of using the cellular network, denoted as *cellular cost*. The second factor relates to the energy consumed in order to use this medium, denoted as *energy cost*. Given the description and structure of the utility function in Eq. 3.5, the cellular and energy costs need to

be converted to actual playback time. That is, the cellular cost is the equivalent of the playback time downloaded through the cellular network, denoted as $U_{A,R}^{C,f\to i}$ for epochs f to i.

With regards to energy cost, an interface supporting radio technologies works in different *power states*, each one related to a different workload as well as power consumption. These states include the *Idle* state, when no Internet connection is required, and consumes the least possible energy. Starting from this state, an interface can be promoted to one or more productive states by spending a fixed amount of energy and time, where productive state means that an interface becomes able to receive or transmit data. Finally, an interface experiences a tail power phenomenon where it stays in a high power state, in anticipation of more data exchange, before returning to its initial *Idle* state.

We denote as $E_{A,R}^{prom}$ and $E_{A,R}^{tail}$ the fixed energy cost of all involved interfaces, of technology A and approach R, when being promoted to a productive state and when experiencing the tailing phenomenon, respectively. Thus, the energy spent until epoch i, $E_{A,R}^{f\to i}$, is:

$$E_{A,R}^{f\to i} = E_{A,R}^{prom} + E_{A,R}^{prod,i} + E_{A,R}^{tail} + E_{A,R}^{f\to i-1}, \tag{3.8}$$

where $E_{A,R}^{prod,i}$ is the energy spent on the productive state of all interfaces for receiving and sharing content during epoch i. Therefore, by identifying the technology, A^*, and approach, R^*, which spend the smallest possible amount of energy until epoch i, $E_{A^*,R^*}^{f\to i}$, we can express the energy cost of a candidate technology, A, and approach, R, in terms of playback time, $U_{A,R}^{E,f\to i}$, by:

$$U_{A,R}^{E,f\to i} = \left(\frac{E_{A,R}^{f\to i}}{E_{A^*,R^*}^{f\to i}} - 1\right) \times L_{A^*,R^*}^{f\to i}. \tag{3.9}$$

The value of $U_{A,R}^{E,f\to i}$ in Eq. 3.9 is effectively the additional content (in terms of playback time) that a user would download if (s)he used technology, A^*, and approach, R^*.

Building on Eq. 3.5, the extended utility function that integrates the energy and cellular costs is:

$$\widetilde{U}_{A,R}^{f\to i} = U_{A,R}^{f\to i} - a_c \times U_{A,R}^{C,f\to i} - a_e \times U_{A,R}^{E,f\to i}, \tag{3.10}$$

where a_c and a_e are the corresponding weights of cell and energy cost, called *cell* and *energy sensitivity*.

In the following, we first discuss the general chunk reception rates over WiFi and cellular connectivity, which are independent of the content retrieval approach (Section 3.2.5). Then we adjust those reception rates to the *Pull* and *PUSH* cases, in Sections 3.2.6 and 3.2.7, respectively. The target is to define the total chunks received until epoch i, that is, the $\lfloor X_{A,R}^{f\to i} \rfloor$ component of Eq. 3.1, which then feeds Eq. 3.2 and eventually the Utility Function defined in Eq. 3.5 and extended in Eq. 3.10.

3.2.5 Chunk Reception Rate over WiFi and Cellular

The size of a single chunk, S, is $S = b \times y$, where b is the bitrate and y is the chunk's playback duration.[2] Then, assuming interference-free cellular downlinks, the cellular chunk delivery rate (per second) will be $X_{cell}(t) = B_{cell}/S$, where B_{cell} is the bandwidth allocated to the user by the cellular network provider. For simplicity, we consider that the cellular bandwidth is stable, irrespectively of the number of active users.[3] We assume that B_{cell} is the bandwidth availability when the train is stopped at some station and that the connection quality deteriorates when the train is moving (*i.e.,* $\ddot{B}_{cell} < B_{cell}$).

Without loss of generality, we assume that the available WiFi access bandwidth per platform of each station is equal to B_{WiFi}, which is equally shared among the users/commuters at each platform. Therefore, the chunk reception rate per second and per user at time t, received through the WiFi AP of a platform is, $X_{WiFi}(t) = \frac{B_{WiFi}}{N(t) \cdot S}$, where $N(t)$ is the number of users requesting Internet access at moment $t \in [t_i, t_{i+1})$ at the given platform. Please note that there is no WiFi connectivity during disconnection periods (*i.e.*, when trains are in-between stations), which means that the chunk reception rate is: $\ddot{X}_{WiFi} = 0$.

In the *PUSH* approach, apart from the chunks received through the Internet, users receive chunks from group members too. The chunk reception rate between group members in the *PUSH* approach is proportional to the group size. Chunks are shared over bandwidth B_{p2p} and the number of chunks that can be shared among a group g of $N_g(t)$ members at a moment t, $X^g_{p2p}(t)$, is defined as:

$$X^g_{p2p}(t) = \frac{B_{p2p}}{N(t) \cdot S} \times N_g(t), \tag{3.11}$$

Eq. 3.11 implies an underlying pseudo-broadcast sharing mechanism (similar to [112]) where a single peer at a time unicasts its content to another peer, while the rest of the members of the group overhear the transmission. Note that the mBaaS platform (mentioned earlier in Section 3.2.1) is also responsible for organising the "sharing turns", given that only one user can unicast at a time.

3.2.6 Pull Reception Rate

In the simple *Pull* approach users *pull* content individually. For cellular Internet access, the total number of chunks received during epoch i is equal to:

$$X^i_{cell,Pull} = |C^i| \times X_{cell} + |\widetilde{C^i}| \times \ddot{X}_{cell}. \tag{3.12}$$

In the WiFi AP case, where the medium is shared between users, the total number of chunks

[2] Note that we do not consider dynamic rate adaptation (*e.g.*, DASH) for simplicity of modelling. We evaluate the proposed model under the lowest possible rate, hence, any higher bandwidth availability will only increase the performance we observe.

[3] In reality, even the cellular bandwidth assigned to each user can be influenced by the level of contention (that is, number of users), but this happens for larger number of users, possibly in the order of thousands, which is not the case of a train (station).

that a user receives through a WiFi AP over epoch i is:

$$X^i_{WiFi,Pull} = \int_{t=t_i}^{t_i+|C^i|} X_{WiFi}(t)dt, \tag{3.13}$$

where t_i is the starting time of epoch i.

Finally, in case users use both interfaces for Internet access, in a *Hybrid* way, the total number of chunks they receive during epoch i is:

$$X^i_{Hybrid,Pull} = X^i_{cell,Pull} + X^i_{WiFi,Pull}. \tag{3.14}$$

3.2.7 PUll and SHare (PUSH) Reception Rate

In the *PUSH* approach, users who belong to a group g of $N_g(t)$ members, at moment t, share their downloaded content. The chunk reception rate from fellow group members is given in Eq. 3.11. Our model does not include multi-hop transmissions which implies that all members of a group have to be within transmission range of each other. For that purpose, we consider WiFi Direct as the technology of choice in order to transmit in long distances with high rates [17]. We also assume that devices can use simultaneously two separate half-duplex WiFi interfaces, one for downloading through the WiFi AP and another one for local sharing. Our approach is also applicable in the simple scenario where only one WiFi interface is available per device, but in that case downloading and sharing should take place sequentially, that is, the users would first download the required chunks and then share them with the rest of the group.

The total volume of content that can be shared between a group (according to Eq. 3.11) is X^g_{p2p}, or more formally, *the theoretical maximum amount of content that can be shared over an epoch i for a group g,* $V^{i,g}_{max(p2p)}$*, is:*

$$V^{i,g}_{max(p2p)} = \int_{t=t_i}^{t_i+|Ep^i|} X^g_{p2p}(t)dt. \tag{3.15}$$

On the other hand, *the maximum volume of content that can be downloaded collaboratively over the connection period of an epoch by access approach A,* $V^{i,g}_{A,C}$*, is:*

$$V^{i,g}_{A,C} = \int_{t=t_i}^{t_i+|C^i|} X_A(t) \times N_g(t)dt, \tag{3.16}$$

where $X_A(t)$ is the number of downloaded chunks per second per group member, during the connection period.

Finally, the corresponding maximum downloaded content under poor connectivity (*i.e.,* using cellular connection when the train is in-between stations), $V^{i,g}_{A,\widetilde{C}}$, is:

$$V^{i,g}_{A,\widetilde{C}} = \int_{t=t_i+|C^i|}^{t_i+|Ep^i|} \ddot{X}_A(t) \times N_g(t)dt, \tag{3.17}$$

where $\ddot{X}_A(t)$ is the number of downloaded chunks per second per group member, during poor connection (or disconnection) period. It follows that *the total volume of content that can be downloaded over an entire epoch i, $X^i_{A,PUSH}$, is:*

$$X^i_{A,PUSH} = V^{i,g}_{A,C} + V^{i,g}_{A,\widetilde{C}}. \tag{3.18}$$

Eq. 3.18 effectively expresses the total number of chunks that the group can download *and* share within epoch i. In case $X^i_{A,PUSH} > V^{i,g}_{max(p2p)}$ the group has downloaded exactly as much content as it can share during epoch i. This would happen when the Internet access bandwidth is higher than the *p2p* one and therefore, users can download faster than they can share. In this case, users continue downloading individually without sharing.

3.3 Quality Assessment Framework Case Study

In this section we perform a case study to demonstrate the usability of the presented framework. At first we analyse an anonymised dataset of London Underground commuters, subsections 3.3.1, before extracting information with respect to group formation potentials for a specific line of the network, in subsections 3.3.2. After that, we present the parameters of the streaming services as well as the network infrastructure we consider in the current case study, in subsection 3.3.3. Lastly in subsection 3.3.4, we evaluate the *Pull* and *PUSH* approaches in the playlist scenario, where each commuter is adding a content to the streaming list with a specific probability. Then, we investigate the *PUSH* approach under a streaming channels scenario, where commuters are engaged with a predefined number of media channels, before concluding with a sensitivity analysis for the cell, energy, and tolerance interval factors. Overall, the following case study covers a variety of streaming delivery methods, in various network access scenarios, which demonstrates the capability of the presented framework to act as a feasibility tool in quantifying and comparing the performance of different approaches in diverse connectivity settings.

3.3.1 Commuter Journey Traces Dataset

We analyse the anonymised commuter trace dataset from London's Oyster Radio Frequency Identification cards. The dataset is a 17-day trace of all journeys that took place in the 11 lines of the London Underground network; the total number of journeys in our dataset is in the order of one million per day. Each journey is identified by an entry and exit point in the network, which is recorded at the granularity of one minute. The traces do not include information regarding the specific lines used by commuters, hence, in order to get an insight of how many passengers are on board a train at a given point throughout the day, we build a custom simulator which takes into account both the entry/exit points of commuters, but also the specific topology of the railway network. That is, we also consider the distance between stations, as well as the intersections of lines at each station and the routes that each line follows.

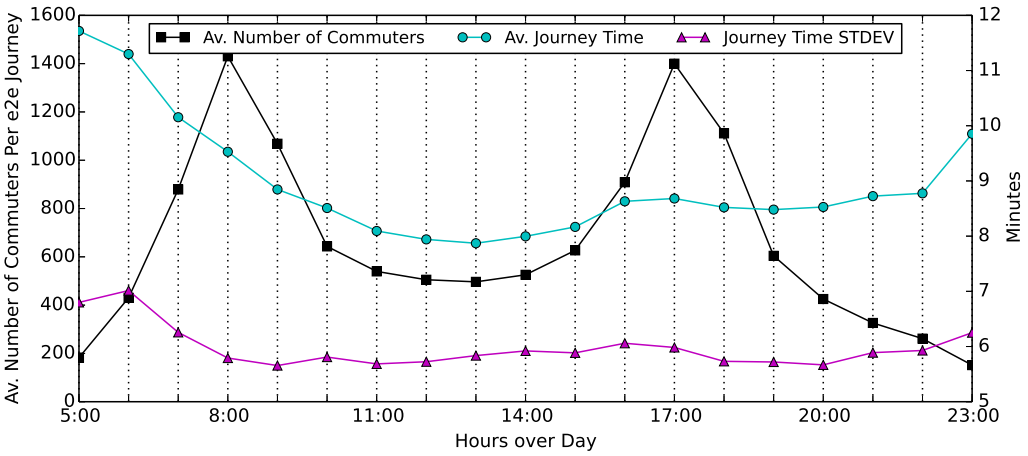


Figure 3.2: Average journey duration and number of commuters per end-to-end journey of the line.

Given that a journey's route might cross more than one lines, and that commuters follow the shortest path with the fewest interchanges between their entry and exit points, we process the trace and assign commuters to specific lines and from there to individual trains.

3.3.2 Group Formation Insights/Potential

Next we focus on a single line to provide insights regarding the commuters' group formation potentials. Fig. 3.2 depicts the average number of commuters that this specific line serves per single end-to-end journey. Obviously, not all commuters travel from one end to the other. The number of commuters is averaged over all trains of the day in hourly granularity. We observe that there are two clear peaks during the morning and the afternoon rush hours. During those times, one train can serve up to 1400 commuters in its end-to-end journey. Depending on their entry and exit points, these commuters can form groups to *PUll and SHare* content from the network.

In order to get a closer view of the potential in forming groups, in Fig. 3.2, we depict the average journey time per commuter along with its standard deviation. According to this plot, commuters use this line on average for approximately 9 minutes with a standard deviation of around 5 minutes. Although the proportion of time that commuters physically share journeys depends on their chosen routes overlap (*i.e.,* entry and exit points), this time is enough to watch or listen to the headline stories of some TV/radio channel.

Clearly, from Fig. 3.2, we see that a line serves different volumes of commuters at different periods of a day, but their distribution over the stations of the line requires further investigation. A more detailed analysis shows that commuters' volumes differ hugely at each station depending on the train direction and the period of the day. We note that the lines normally start and terminate at the outskirts of the city centre, but always cross the city centre itself. In Fig. 3.3 we present the average number of commuters per train at three different stations over the duration of a day. The

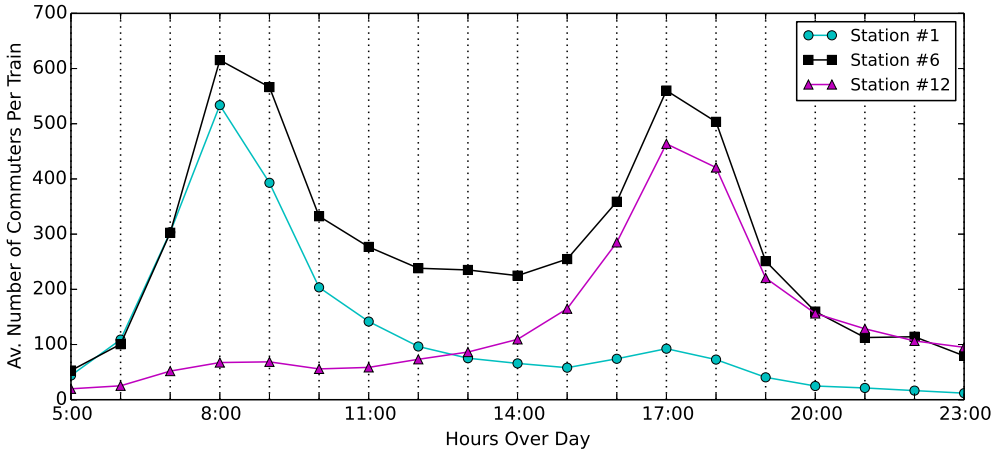


Figure 3.3: Number of commuters per train (single direction) for a station in the beginning, middle, and end of the line.

Standard Setting Variable	Value
Total bandwidth per station, B_{WiFi}	0.5 Gbps
Cell rate - moving ($\ddot{B}_{cell}$)/static (B_{cell}) train	150-550 Kbps
Sharing bandwidth, B_{p2p}	54 Mbps
Connection period duration for each station i, $\|C^i\|$	20”
Participation probability, p_{list}	50%
Music/Video bit-rate	160/419 Kbps
Chunk playback time, y	5”
Playlist duration, Y	15’
Zipf’s exponent, a	1
Playlists generated per station, Z	5-95
Delay (a_d), Energy (a_e), Cell (a_c) Sensitivity	3, 0, 0
T_d	5”

Table 3.2: Quality Assessment Framework Default Evaluation Setting

three stations chosen are towards the beginning of the line (Station 1), the middle of the line (a central location, Station 6) and the end of the line (Station 12). Interestingly, and somewhat expectedly, we see that the station at the beginning of the line serves most users in the morning rush hours (*i.e.,* users move towards the city centre), whereas the station towards the other end of the line serves most users during the afternoon rush hours (*i.e.,* on the way back). The station in the city centre presents two peaks one in the morning and one in the afternoon rush hour.

This result makes clear the need for provisioning of Internet streaming services according to journey patterns, station locations, and the specific hour of a day. That is, given that connectivity is available only when trains are within stations, where they normally stay for about 20-30 seconds and that disconnection periods last 1-4 minutes, the amount of content that needs to be buffered in order to avoid playback disruptions differs hugely. Furthermore, since different stations are busy at different periods of day depending on the area and the direction of the train, the system has to be carefully modelled to include these factors in order to guarantee undisrupted playback.

3.3.3 Simulation and Evaluation Setup

In our setting the users have access to both WiFi connectivity when trains are in stations and cellular connectivity throughout their journeys. Despite the fact that in the case of the London Underground cellular connectivity is not available, here we include this access option for completeness. We also assume that users can use both their cellular and their WiFi interfaces simultaneously to receive content (*Hybrid* approach).

Although the current WiFi access deployment at the London Underground network provides download speeds of up to 100Mbps, here, we consider a future, overprovisioned network where each station is connected to the Internet by 500Mbps links. This bandwidth is split between all platforms and among all lines passing through each station.

For the cellular case, we assume that users get between 150Kbps and 550Kbps - closer to the lower value when the train is moving and to the highest one when the train is static in the station. Finally, we set the users' sensitivity to delay (a_d in Eq. 3.5) equal to 3, which means that for every second of disruption the users' utility decreases by three, whereas their utility increases by one for every second of undisrupted playback. This value is specifically chosen as an extreme scenario, where users are not tolerant to disruptions.

In addition to the different access methods, we evaluate the users' QoE (*i.e.,* Eq. 3.6) when two different types of content are available, that is, music content (at 160Kbps) and video content (at 419Kbps). The chosen bit rates are the lowest possible for streaming of acceptable quality. In cases where the system can achieve minimum disruptions, an adaptive increase of the corresponding bit rates can be applied (*e.g.,* through DASH), but this is out of the scope of this study. Unless mentioned otherwise, the full list of settings of our evaluation setup is given on Table 3.2.

Due to the limitations of the environment under investigation, we consider that users cannot individually choose to stream content and only if requests match, then users form groups. This would clearly result in few and small groups, effectively reflecting the *Pull* case. Instead, we evaluate two specific scenarios. In the first one, we assume that users create music or video "playlists" according to genre preferences, *e.g.,* sports clips, or jazz music. Users then join a group and add their own preferences to the list. We assume that between 5 and 95 playlists are generated in each epoch, resulting in more than 500 playlists available throughout a train's end-to-end journey. The playlists' duration is set to 15 minutes to reflect the average journey time plus standard deviation. In our second scenario, users tune in to radio or TV channels [13], [15]. In particular, we assume that 50 channels are supported by the system, which would cover the main broadcasters of a region. The difference between the *playlist* and *channel* scenario is that in the case of the latter, users form larger groups. In both cases, we consider that users select a playlist or channel to subscribe according to a Zipf distribution [3].

Note that we consider that both the available channels and the created playlists are reccomended to the rest of the commuters who are not participating in any other group at each station. A commuter

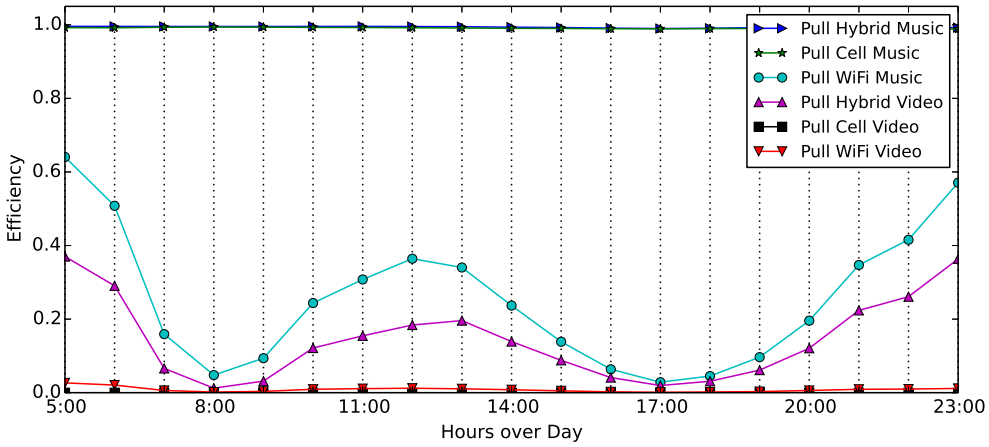


Figure 3.4: 15-min Playlists scenario *Pull* efficiency for WiFi, Cell, and Hybrid access methods.

is interested in this list and participates with a probability p_{list} while choosing the r-th element of the list according to a Zipf distribution with probability $f(r;a,Z)$ where a is the Zipf exponent and Z is the number of elements in the list.

We measure the users' perceptual QoE as expressed through Eq. 3.6. The initial tolerance interval (T_d in Eq. 3.6) is set to 5 seconds, after the expiration of which each user abandons the attempt to receive content. This setting is based on studies that assess the patience of users on web content loading [85].

3.3.4 Performance Evaluation

Pull Approach - Playlists scenario

Fig. 3.4 illustrates the average (over all trains) efficiency of *WiFi*, *Cell*, and *Hybrid* access methods of the *Pull* approach for music and video streaming over one operational day of the tube network. Given that with the *Cell* approach each user experiences speeds between 150Kbps and 550Kbps and that the bitrate for streaming music is 160Kbps, it is straightforward that the network can support such a service. Hence, the *Cell* approach (and stemming from that, obviously, the *Hybrid* approach) can provide a satisfying QoE. On the other hand, the *WiFi* access method experiences disruptions which increase as the volume of commuters increases (*i.e.,* during rush hours - see Fig. 3.2 and Fig. 3.3). In the case of video streaming all three access approaches, *WiFi*, *Cell* and *Hybrid* perform worse than music streaming. In the following we do not depict the music bitrate for the cell access approach, as it clearly can be supported when cell access is available (not for the case of the London Underground network though which solely relies on WiFi).

PUSH Approach - Playlists scenario

From Fig. 3.5 we observe that streaming music over the WiFi network is challenged by disruptions/disconnections, achieving performance close to 0.8 at best, while during rush hours this can go

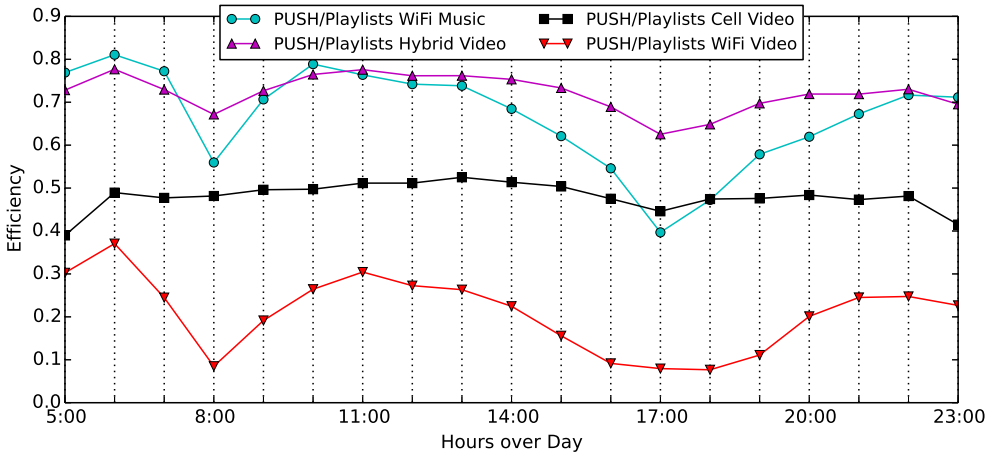


Figure 3.5: 15-min Playlists scenario *PUSH* efficiency for WiFi, Cell, and Hybrid access methods.

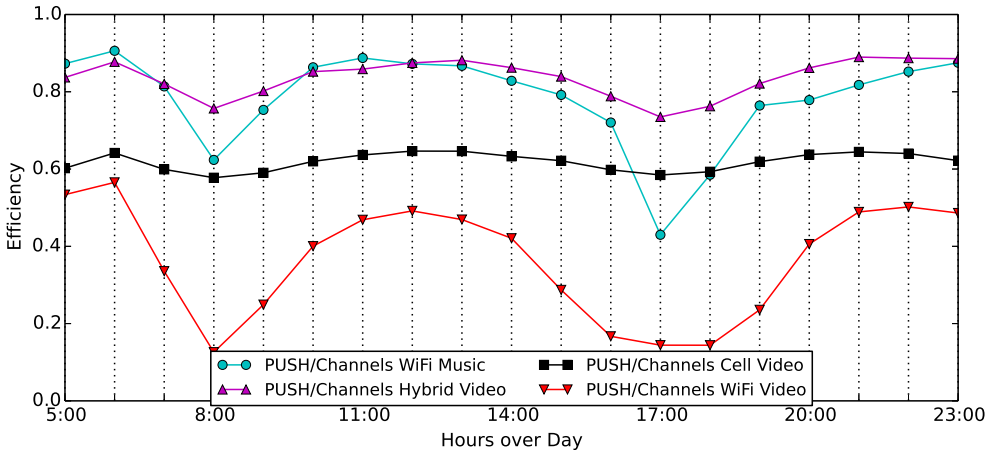


Figure 3.6: Streaming Channels - *PUSH* efficiency for WiFi, Cell, and Hybrid access methods.

as low as 0.4. In the case of Video playlists, which is even more demanding (*i.e.,* higher bit rates), achieves lower performance ranging between 0.7-0.8 when utilising both the WiFi and the cellular connection. Note that the performance of *PUSH* approach is acceptable during off-peak time, especially considering the extreme delay sensitivity assumed here. Overall, based on the parametrisation of our setting, collaborative download can provide acceptable service quality, especially when it is combined with cellular connectivity. In the following we experiment with different parameters to investigate the further improvement of the quality.

PUSH Approach - Channels scenario

Efficiency performance follows similar trends in the case of the second scenario (*i.e.*, TV/radio channel streaming - see Fig. 3.6), where, however, we observe a small increase in the QoE perceived by the users. This is because, in the "channel" case, groups are formed for longer time periods

[189] while additional users join the group as trains move towards their destination. That is, groups are bigger in size compared to the playlists scenario, giving the opportunity to move more content locally. Clearly, during the rush hours, the available WiFi bandwidth per commuter is significantly decreasing and, despite the size of the formed groups, the efficiency is still quite low since each user can download and share a very small portion of the desired content. As we show next, it requires a large amount of available WiFi bandwidth at each station in order to increase the performance of the system. Furthermore, another important factor that affects performance is the initial tolerance interval as we show later in this section.

PUSH Approach - WiFi Bandwidth Factor

In general, more available bandwidth at each platform/station improves performance; however, here we examine whether investing more on bandwidth would pay off in terms of users' QoE. In Fig. 3.7 we present the case for video streaming to groups of commuters (*i.e.*, *PUSH* approach) for the WiFi and the Hybrid access methods.

Interestingly, the efficiency achieved by the WiFi access method during peak times exceeds the one achieved during off-peak when the available bandwidth at each station is significantly large ($\geq$ 1Gbps). In these cases, a larger number of commuters at each train results in larger groups. In turn, each group gets a larger stake of the available bandwidth. Overall, we see that even when 1Gbps is readily available at each of the hundreds of tube stations throughout the London Underground network, it is still difficult to achieve uninterrupted video streaming relying on the WiFi connectivity alone, even at off-peak times. On the other hand, when collaborative streaming is combined with cellular connectivity, we observe a performance close to 0.8, even for smaller bandwidth values and rather disruption-sensitive users (*i.e.,* $a_d = 3$).

PUSH Approach - Cell and Energy Sensitivity

The cell sensitivity integrates the cost factor, that is, the monetary cost to download through the cellular network (*i.e.,* a_c in Eq. 3.10), as opposed to the WiFi access. In general, the efficiency decreases linearly in all the retrieval approaches (*i.e., Pull* and *PUSH*) with respect to the cell sensitivity. In more details, in the *Pull* case it declines with exactly the same rate during both peak and off-peak times, since the amount of data that each commuter downloads from the cellular interface in each case remains the same. On the other hand, for the *PUSH* approach (Fig. 3.8) the efficiency decline rate is less steep and the difference between the peak and off-peak time is proportional to the average group size. In this approach, users also exchange a significant amount of data minimizing the data to be directly downloaded from the cellular network.

We use the power state machine presented in [71] to evaluate the energy sensitivity of the cellular and WiFi interfaces of smartphone devices for each one of the power state (*i.e.,* Promotion, Productive and Tail). Our findings show that despite the fact that *PUSH* uses an additional interface for sharing data, the low promotion and tail energy required by the sharing interface decreases the

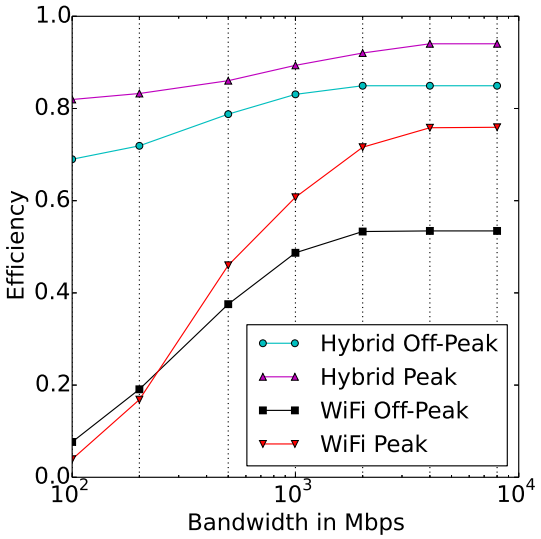


Figure 3.7: 15-min Video Playlists scenarion *PUSH* Efficiency for WiFi and Hybrid access methods for increasing bandwidth (Peak time: 8am, Off-Peak: 11am).

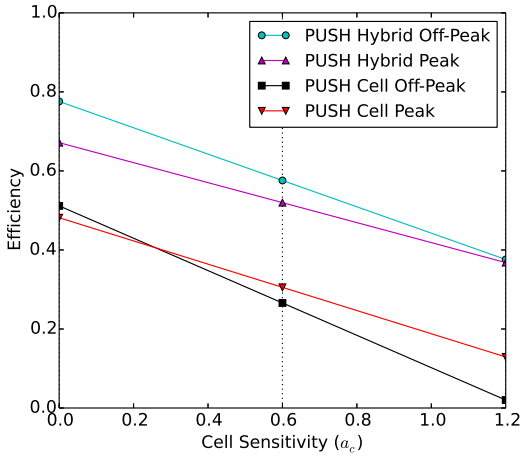


Figure 3.8: Cell Sensitivity Increase - Scenario 1: Video Playlist

overall performance as we increase the energy sensitivity. Finally, we notice that increasing the sharing energy transmission coefficient, a_{tr}, causes only slight decline in the performance proportional to the coefficient's actual value (Fig. 3.9). This is partly because local transfers (through the sharing interface) complete much faster, therefore, the sharing interface spends little time in "transmission mode".

PUSH Approach - Tolerance Interval

The *"Initial Tolerance Interval"* of users indicates the patience of users to startup delay. Throughout our evaluation, this interval was set to 5 seconds, according to related studies on users' tolerance to

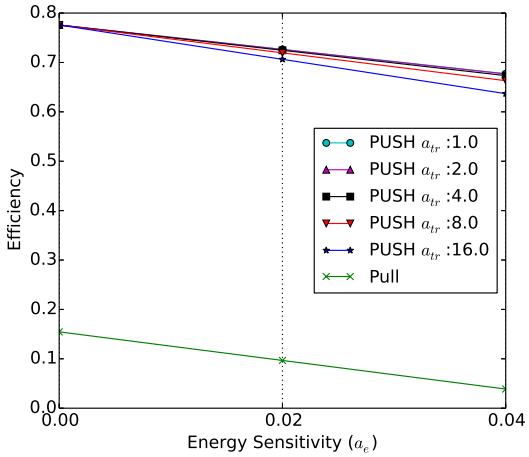


Figure 3.9: Energy Sensitivity Increase - Scenario 1 Video Playlist, Hybrid

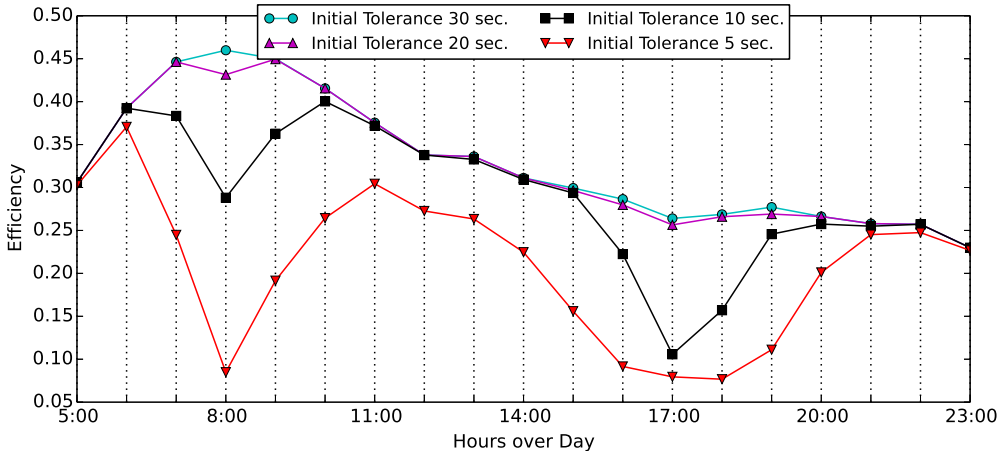


Figure 3.10: 15-min Playlists scenario PUSH- WiFi Efficiency for different Initial Tolerance Intervals.

startup delays [85]. In this experiment, we investigate the effect of the tolerance interval on users' QoE (in Fig. 3.10); we assume a 15-min playlist when users are pulling and sharing content over WiFi, and the tolerance interval is set to 10, 20 and 30 seconds. As the users' tolerance increases, we observe that the QoE increases too. This is a straightforward result, given that the groups formed in this case are larger and can therefore get larger share of the available WiFi bandwidth. Expectedly, we observe that after the 20-second threshold the performance does not improve any further.

3.4 Conclusions

In this chapter, we presented a quality assessment framework that characterises the QoE of media streaming in urban railway networks, as a realistic setting for collaborative content downloading under disrupted connectivity settings. The assessment framework quantifies the utility of the streaming

application delivery by taking into account application parameters (such as the number of channels, content bitrate *etc.*), connectivity patterns (*e.g.,* connection and disconnection periods, existence of cellular connectivity *etc.*), and user subjective parameters (like playback disruption tolerance, energy as well as cellular cost sensitivity). As a case study, we analysed commuters' traces of London Underground before applying our model to their mobility patterns. We found that it is difficult to maintain undisrupted playback, especially in case of high bit rates, *i.e.*, video content, but at the same time, well thought-out collaborative mechanisms can increase the perceived QoE even under such a challenging conditions. When cellular connectivity is available, performance improves considerably, given that users can utilise both interfaces (the WiFi and the cellular one) simultaneously.

We argue about the importance of such a quality assessment framework in deploying LLAs since it can work as the basis for incentivising users, *i.e.,* if you participate in the group, your quality will be improved by $X\%$ (direct incentive) or your contribution is estimated to Y dollars (indirect incentive). Furthermore, AppSPs can exploit such a framework as a feasibility study tool in order to parameterise their LLA for maximising their performance.

Chapter 4

Enabling LLAs in Mobile Connectivity Settings

4.1 Introduction

In this Chapter, we investigate the problem of *stateful LLA* provisioning, under the assumption that the VM of an LLA can migrate to another location within a reasonable timeframe, in the scenario of *mobile users*. In particular, we assume a set of geo-distributed cloudlet infrastructures, each of which is owned by an INCP that is willing to lease its resources to host LLAs. On the other hand, AppSPs are willing to pay INCPs in order to run their applications in the edge infrastructure. Therefore, AppSPs provide higher QoS to their customers, while INCPs are incentivised to both maintain and expand their edge infrastructure. In this setting, in order to support dynamic LLA provisioning for mobile users, we have to address the interdependent problems of:

1. Cloudlet monetary profit generation at the edge, *i.e.,* how INCPs make money,
2. Real-time cloudlet resource management, *i.e.,* which LLA should be deployed and where.

We assume interactive applications with strict deadlines (in the order of 10-50 ms, as in the case of augmented reality [55], for example) and users that move between cells. Similarly to cloud systems, the computing resources of the cloudlets are available in the form of Virtual Machines (VMs) and are located *i)* at base-station cells and *ii)* at middle-tier network locations. Provisioning of resources at base-station cells (hexagons closest to the edges in Fig. 4.1) is particularly challenging due to the mobility of users, which we mainly focus on in this here. The back-end clouds serve as the *final-call* serving location (Fig. 4.1) for user requests that fail to obtain resources at the cloudlets.

In this chapters, we argue for a market-based solution that brings together INCPs who lease their resources, and AppSPs which are interested in renting edge resources to improve the QoS of their mobile users. To that end, we propose a design for a market mechanism tailored to the mobile users demand dynamics that benefits both LLAs' QoS and INCPs' income. We introduce *Edge-MAP*, as the underlying framework that enables the *in-network, on-demand provisioning market*

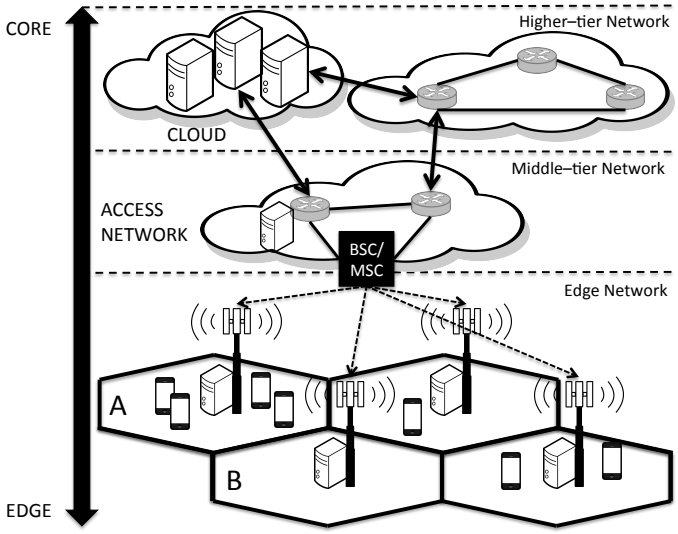


Figure 4.1: Cloudlet Infrastructure Deployment

for LLAs. A distinct novelty of Edge-MAP, is that it perceives the INCPs' resources as a *"pool of interconnected virtualised hardware"* offered via independent marketplaces located at each cell. According to this view of the network, resources physically located but under-utilised at some cell, *e.g.,* cell B in Fig. 4.1, can be advertised in neighbour cell markets where demand exceeds supply, *e.g.,* cell A in Fig. 4.1. We assume that a Vickrey-English-Dutch (VED) auction [30] is deployed at each cellular market to provision LLAs' instances over the offered VMs in polynomial time. Edge-MAP fosters competition among INCPs (located at various cells) by providing them profit-making opportunities in order to offer their (unused) resources at marketplaces of both local and remote cells.

The main technical contributions of this chapter are as follows:

1. We introduce the Edge-MAP mechanism for the provisioning of LLAs over a distributed INCPs' infrastructure, tailored to the challenging scenario of mobile users.
2. We apply VED auctions in problems where the demand/supply conditions evolve over time.
3. We evaluate Edge-MAP on realistic vehicle traffic patterns.

4.2 Edge-MAP Design and System Model

In this section, we present Edge-MAP's design principles, that support *on-demand provisioning markets for mobile users*, followed by the system model description.

4.2.1 Edge-MAP Design Principles

LLA provisioning over geo-distributed cloudlet resources differs from provisioning in the cloud in that the allocation of cloudlet resources has to take into account the impact of the network conditions, *i.e.,* the latency between the end users and the cloudlets' points of presence [20], on applications'

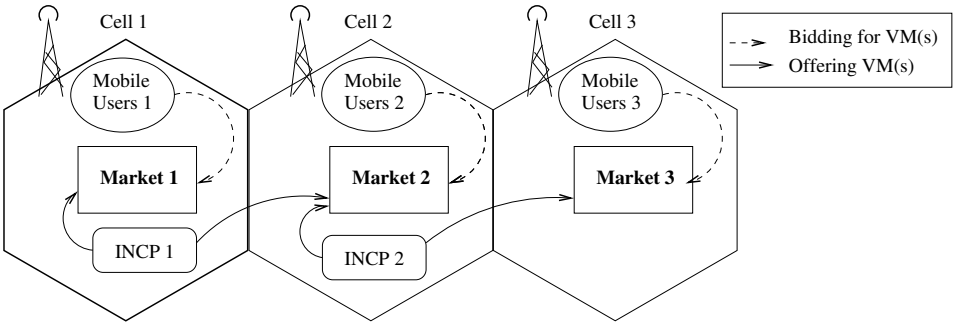


Figure 4.2: VM offerings by the INCPs and user biddings for VMs in three adjacent cells.

QoS. Moreover, in the case of mobile users, the latency to an allocated VM changes as soon as users handover to different base-stations. Hence, even optimal VM allocations to user requests get outdated over time; that is, users' mobility/handoffs should be followed by VM reassignments (when required), referred to as *VM handoffs* in [163]. Instead, under static provisioning, where a static number of VMs are allocated to an application for long periods of time, VM handoffs might lead to idle VMs that could be used by other applications. In this work, in order to avoid the underutilisation of VMs, we argue for the need of *on-demand LLA provisioning*, where a VM for an application is instantiated *upon an end-user request* for the duration of the end-user's engagement (or handoff to another cell).

A simplified version of the on-demand LLA provisioning process, described in detail in section 2.2, consists of *i*) discovery of available resources (resolution), *ii*) resource allocation, and *iii*) resource configuration (*i.e.,* booting up of VMs and memory page migration) [74]. In order to obtain a responsive and efficient provisioning system for mobile users, we focus on minimising the amount of time spent for resource discovery and resource allocation. With the latest advances in virtualisation technology, the configuration overhead is greatly reduced. For instance, Unikernel VMs boot in typically less than 150 ms [125]. The VM configuration overhead is expected to further reduce in the future, and therefore, *we focus on reducing the time overhead of discovery and allocation of resources*.

In the competitive setting of INCPs that we envision, the discovery and allocation of resources (*i.e.,* VMs) happen through auction mechanisms that take place in individual markets [98, 174], each located in a base-station cell. In a cell's market, the offered VMs are considered as *items*, while the mobile users connected to the corresponding cell are *bidders* that wish to acquire at most a single VM/item. The auction's purpose is to derive the price vector as well as the item-bidder assignments that characterise a *competitive equilibrium*. That is, the auction has to: *i*) satisfy bidders' demand for the given price vector, and *ii*) fully allocate every item with a positive price, *i.e.,* every unallocated item's price is 0.

A distinct feature of Edge-MAP is that INCPs can offer their VMs in other cells' markets with

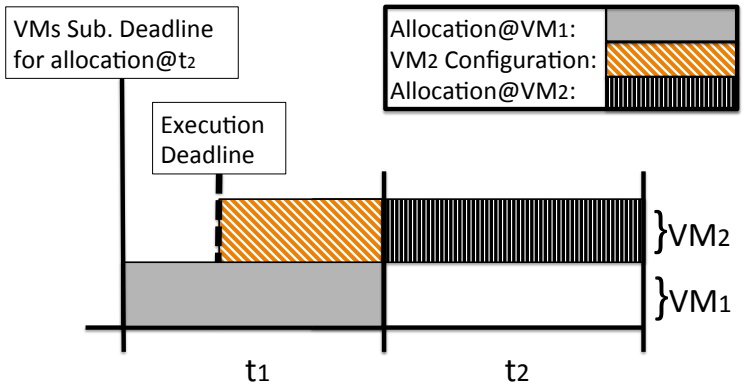


Figure 4.3: Time-slot t_1 duration decomposed into *i*) the auction execution deadline, *i.e.,* the maximum expected execution time of the auction, and *ii*) the VM configuration overhead for provisioning at time-slot t_2, *e.g.,* VM_1 handoff to VM_2 at t_2.

the condition that a particular VM can only participate uniquely to a single market at any point in time. Being able to offer VM resources to adjacent cells' users leads to profit opportunities for INCPs. As we explain in detail later in Section 4.3, INCPs discover such opportunities through a feedback system. As an example, consider Fig. 4.2, where three adjacent cells numbered 1–3 are depicted with their connected mobile users. There is a single INCP in both cell 1 and cell 2 and no INCPs in cell 3. In each cell, there is a market dedicated to its local users to discover and allocate available VMs. INCP 1 of cell 1 offers a proportion of its VMs to its local market (Market 1) and the remaining VMs to second cell's market (Market 2). Similarly, INCP 2 offers part of its VMs to its local market (Market 2) and the remaining ones to the adjacent third cell's market (Market 3). The connected users of each cell can only bid in their local market, which simplifies the discovery process.

Provisioning of VMs happens through periodic/discrete-time execution of auction mechanisms, where the minimum duration of each period/time-slot is restricted by the aggregated time overhead of auction execution and VM configuration, as we show in Fig. 4.3. In each period, the bidders adjust their demand subject to their local cell, and the associated network conditions, while the INCPs adjust their supply, in terms of offered VMs, subject to their current utilisation. That is, the objective of minimising the time overhead of resource discovery and resource allocation is equivalent to *minimising the time of accessing a market and execute an auction*.

Along these lines, the proposed Edge-MAP mechanism relies on VED auctions [30] for on-demand provisioning for the following reasons:

- VED auctions *derive the unique minimum competitive equilibrium prices* [168], known as Vickrey-Clarke-Groves (VCG) prices; that is, the bidders cannot acquire their assigned items for a lower price in any other competitive equilibrium.

- The VED auctions result in the VCG prices *starting at any possible initial price* for each one of the items.

Because the Edge-MAP is executed repetitively, a VED auction can re-use previously found VCG prices and reduce the execution time of the mechanism, *e.g.,* if the supply and demand conditions of the market remain the same, the VCG prices of the new time-slot will be identical to the previous one, and the auction will terminate immediately since it starts from its equilibrium prices. Moreover, the bidders are *truthful* since they acquire an item in its minimum possible price; that is, they do not have incentives to deploy complex strategies that would prolong the execution of the auction.

Finally, in Edge-MAP instead of considering a single centralised market, *we approach each cell as a distinct marketplace, organised by a local auctioneer to serve the on-demand provisioning requests of its currently connected users*. The choice of cell-based markets comes with the advantages of:

- *Resource discovery and allocation time overhead minimisation*, since a user request is accessing a market immediately at her local cell. At the same time, the auction execution involves a considerably smaller number of bidders, leading to a significantly lower execution time.

- *Providing profit opportunities to INCPs*, which can offer their VMs to different cellular markets at different prices. In particular, Edge-MAP provides feedback to INCPs regarding their demand in each market, fostering their competition as we explain in 4.3.2.

Edge-MAP Overview: Our design, on top of the already introduced AppSPs and INCPs, involves the following entities:

- *Auctioneer:* The entity that collects arriving bids for LLA provisioning and allocates INCPs' VMs to the highest set of bids. There is *exactly one auctioneer per cellular market*.

- *AppSP Agents:* The entities that represent AppSPs in every cellular market. These entities actually bid on behalf of the mobile users for VMs offered in each market, with respect to the requested LLA requirements.

In Fig. 4.4, we depict the Edge-MAP's provisioning process upon the arrival of a new user in a cellular market. At first, the mobile user sends an application request to the local market which is directed to the corresponding AppSP Agent (step a). The AppSP Agent is aware of the application requirements, *i.e.,* in terms of how the latency affects its QoS. In particular, we assume that the AppSP has pre-estimated the potential gain in QoS that a VM provisioning can produce at each INCP for a specific LLA request arriving at the local cellular market. Since the VED auction is an incentive-compatible mechanism [30], the AppSP Agent acts truthfully and bids for resources by just setting the bids equal to the actual gain, in terms of QoS, of the application (steps b and c). Next,

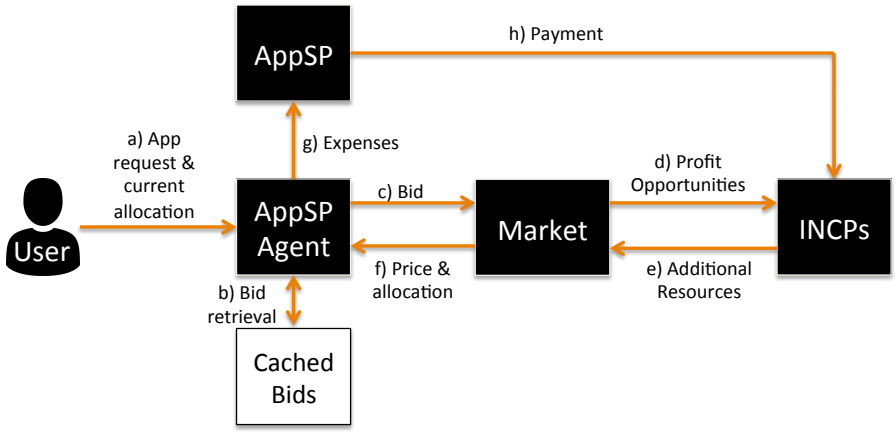


Figure 4.4: Edge-MAP Overview

the VED auction is executed followed by the INCP's feedback phase about profit opportunities (step d) that aim to attract more VM resources to the market (step e); concluding in the VM provisioning for the user at a specific price for the next time-slot utilisation (step f). Finally, the AppSP Agent periodically informs the AppSP for the expenses of this market (step g) in order to arrange the payment of the INCPs (step h).

4.2.2 System Model

We consider the state of a cellular market before the execution of the auction mechanism, *e.g.,* the beginning of period t_1 in Fig. 4.3, that will define the LLA provisioning for the next time-slot. Let $\mathscr{S} \triangleq \{1,2,3,...,S\}$ be the set of LLAs, where each LLA s is requested by $\mathscr{M}_s$ *unit demand* users, *i.e.,* interested in *at most* one VM, that are currently connected to the cell of the market. We denote by $\mathscr{M} \triangleq \{1,2,3,...,M\}$ the set of all mobile users, *i.e.,* $\mathscr{M} = \bigcup_{s\in\mathscr{S}} \mathscr{M}_s$, that *bid* for $\mathscr{H} \triangleq \{1,2,3,...,H\}$ VMs, offered at the price vector $p \triangleq (p_h : \forall h \in \mathscr{H})$. The VMs are submitted to the market by a set of $\mathscr{C} \triangleq \{1,2,3,...,C\}$ INCPs, where each $c \in \mathscr{C}$ contributes $\mathscr{H}_c$ VMs, such that $\mathscr{H}_c \subseteq \mathscr{H}$ and $\mathscr{H}_c \cap \mathscr{H}_{c'} = \emptyset$ for all $c, c' \in \mathscr{C}$ when $c \neq c'$.[1]

We assume that each AppSP Agent is aware of the latency between the current cell and the location of nearby INCPs. Based on the latency information, the AppSP Agent of a LLA $s \in \mathscr{S}$, derives the expected QoS produced by provisioning a VM at INCP $c \in \mathscr{C}$ for serving an s LLA request for the next time-slot, $u_{s,c}$. Furthermore, assuming an always available default provisioning (*i.e.,* back-end cloud) infrastructure for LLA s, the AppSP Agent can similarly estimate the default expected QoS, $u_{s,\emptyset}$. Then, the valuation (*i.e.,* QoS gain) of a user $m \in \mathscr{M}_s$ with respect to a VM $h \in \mathscr{H}_c$ is:

$$v_{m,h} = u_{s,c} - u_{s,\emptyset}. \tag{4.1}$$

Note that $v_{m,\emptyset} = 0$. The notation used throughout the Chapter is given in Table 5.1.

[1] Each VM is offered to a cellular market uniquely.

$\mathscr{S}$	Set of LLAs.
$\mathscr{M}, \mathscr{M}_s$	Set of users connected to the cell, Users requesting LLA s.
$\mathscr{C}$	Set of INCPs.
$\mathscr{H}, \mathscr{H}_c$	Set of offered VMs, Set of offered VMs by INCP c.
$\bar{\mathscr{H}}$	Universally allocated VMs.
p, p_h	Vector price of offered VMs, Price of VM h.
$u_{s,c}$	QoS produced by serving a request of LLA s at INCP c.
$u_{s,\emptyset}$	QoS produced by serving LLA s at its default location.
$v_{m,h}$	Valuation of user m for VM h.
$D_m(p)$	Demand correspondence of user m function to price vector p.
$\tilde{S}, S^*, E^*$	Set of VMs in positive excess demand, excess supply, and excess demand.
p^{VCG}	VCG price vector of offered VMs.

Table 4.1: Edge-MAP Notation

4.3 Edge-MAP Mechanism

In this section, we introduce Edge-MAP's micro- and macro- level operating components of:

1. **Edge-MAP Cellular:** That is deployed in each cellular market to perform the on-demand provisioning of LLAs.

2. **Edge-MAP Orchestrator:** That provides feedback to each INCP regarding profit opportunities of its over-demanded VMs in each cellular market.

4.3.1 Edge-MAP Cellular Operation

We describe Edge-MAP's micro-level operating component by focusing on a single cellular market. At first, we define the set of over-demanded/supplied VMs that characterise any *Multi-item auction with unit demand bidders*. After that, we provide details about how VED auctions derive the VCG equilibrium by increasing (decreasing) the price of the VMs that are considered over-demanded (over-supplied) in the context of Edge-MAP mechanism.

VMs in Excess Supply and Excess Demand

We consider the extended market of $\mathscr{H}^* = \mathscr{H} \cup \{\emptyset\}$ VMs, where the *null item* $\{\emptyset\}$ corresponds to the default VM provisioning of each request at the corresponding back-end cloud.[2] Let p_h be the price of VM $h \in \mathscr{H}^*$ and $v_{m,h}$ be the valuation of bidder/user m for VM h (Eq. 4.1). The *demand correspondence* of user m is defined as:

$$D_m(p) \triangleq \{h \in \mathscr{H}^* : v_{m,h} - p_h \geq v_{m,h'} - p_{h'}, \forall h' \in \mathscr{H}^*\} \tag{4.2}$$

in other words, the $D_m(p)$ set includes the VMs that maximise the user's valuation after the price reduction, known as net-valuation, *i.e.,* $v_{m,h} - p_h$. Note that, at the default provisioning location, the VMs' price equals to zero, *i.e.,* $p_\emptyset = 0$. In Fig. 4.5, a market of four VMs and four users is depicted forming the following demand correspondence sets: $D_{m_1}(p) = \{h_1\}$, $D_{m_2}(p) = \{h_2\}$,

[2]The null item is allocated to requests as many times as necessary so there is no request left unserved, *i.e.,* without an assigned VM.

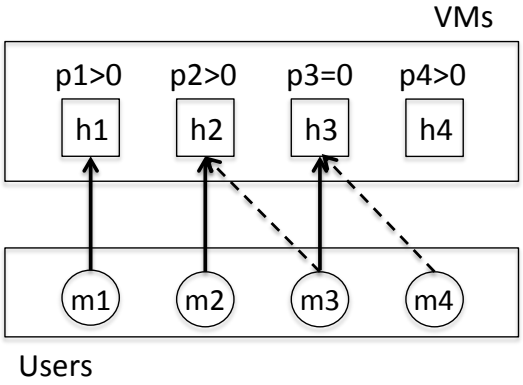


Figure 4.5: VMs assignment example.

$D_{m_3}(p) = \{h_2, h_3\}$, and $D_{m_4}(p) = \{h_3\}$.[3] We use the example of Fig. 4.5 as a point of reference in the upcoming definitions.

Given the users' demand correspondence to a price vector, the *universally allocated items*, $\tilde{\mathscr{H}}$, are defined as the set of VMs which either have a price equal to 0 or satisfy at their current price at least 2 bidders, *i.e.,* $p_h = 0$ or $\exists m, m' \in \mathscr{M} : h \in D_m(p) \cap D_{m'}(p)$ where $m \neq m'$ for each $h \in \tilde{\mathscr{H}}$. That is, in the example of Fig. 4.5, the set of universally allocated items is $\tilde{\mathscr{H}} = \{h_2, h_3\}$. Then for a set of universally allocated items, authors in [30] define the set of *positive excess demand*, $\tilde{S}$, as the universally allocated items/VMs with positive price, *i.e.,* $\tilde{S} \triangleq \{h \in \tilde{\mathscr{H}} : p_h > 0\}$. In the example of Fig. 4.5 for $\tilde{\mathscr{H}} = \{h_2, h_3\}$, the positive excess demand set is $\tilde{S} = \{h_2\}$ since $p_{h_2} > 0$ while $p_{h_3} = 0$. Furthermore, they prove that set $\tilde{S}$ can be identified in polynomial time by the FindUnivAllocItems procedure presented in [162].

On the other hand, the set of *excess supply*, S^*, is defined as the set of *not* universally allocated items/VMs with positive price, *i.e.,* $S^* = \{h \in \mathscr{H} : p_h > 0\} \setminus \tilde{S}$; meaning that, in the example of Fig. 4.5, the excess supply set is $S^* = \{h_1, h_4\}$ since both VMs have a positive price while not belonging to the set of positive excess demand. Finally, the concept of the *excess demand* set is introduced. Intuitively, a set of VMs E is in excess demand at a given price vector, if *i*) the number of VMs in each proper subset T of E, $T \subset E$, is strictly smaller than the number of users that demand a VM in T, and *ii*) the users that demand at least a VM in E do not request VMs outside E. Furthermore, in [29], the authors prove that there exists a unique set in *excess demand with maximal cardinality* E^* that can be identified in polynomial time by using the "Ford and Fulkerson" algorithm, presented in [82]. In the example of Fig. 4.5, $E^* = \{h_2, h_3\}$.

VCG Equilibrium Derivation

In this section, we explain how the Edge-MAP mechanism applies VED auctions at a micro-level to reach the VCG equilibrium of the cellular market. VED auctions can derive the VCG equilibrium prices, p^{VCG}, as well as the corresponding users to VMs assignment, *i.e.,* $x^{VCG} : \mathscr{M} \rightarrow \mathscr{H}$, starting

[3] We do not depict the null element.

from any initial price vector p.[4] At a given time-slot, *e.g.,* time-slot t_2 in Fig. 4.3, Edge-MAP exploits VED auctions by initialising the price of each VM in the market according to the p^{VCG} solution found in the previous time-slot, *e.g.,* t_1. At the core of VED auctions is the elimination of the VMs in excess demand, E^*, and excess supply, S^*, since they are associated to the price vector p^{VCG} by the following theorem, proved in [135]:

Theorem 1. A price vector, p, equals the VCG price vector, p^{VCG}, if and only if the sets of excess demand and excess supply are empty, *i.e.,* $E^* = \{\emptyset\}$ and $S^* = \{\emptyset\}$.

Therefore, VED auctions eliminate sets E^* and S^* iteratively by updating the price of each VM h at the k-th iteration, p_h^k, according to:

$$p_h^k = \begin{cases} p_h^{k-1} + \Delta p, & if\ h \in E^{*,k-1}, \\ p_h^{k-1} - \Delta p, & if\ h \in S^{*,k-1}, \\ p_h^{k-1} & otherwise. \end{cases}$$

where if $h \in E^{*,k-1}$ (*i.e.,* eliminating the excess demand set upon iteration $k-1$), the price is increased by Δp; on the other hand, if $h \in S^{*,k-1}$ (*i.e.,* eliminating the excess supply set upon iteration $k-1$), the price is decreased by Δp. The policy of increasing the prices of each VM in E^* is known as E^*-*increase* while the policy of decreasing the prices in S^* is known as S^*-*decrease*.

Lemma 1: Consecutive E^*-*increase* (S^*-*decrease*) policy applications eliminate the excess demand E^* (excess supply S^*) set in polynomial time.

Proof. There is a maximum price that any VM in the market could be allocated $\bar{p} = \max_{\forall m \in \mathcal{M}, h \in \mathcal{H}} v_{m,h}$. Thus, the E^*-*increase* policy takes at most $\lceil \bar{p}/\Delta p \rceil$ steps to exclude every VM in E^* from the users' demand correspondence, by increasing their price as their net valuation $v_{m,h} - p_h$ declines; resulting, in the elimination of E^*. Similarly, for the S^*-*decrease* policy, it takes at most $\lceil \bar{p}/\Delta p \rceil$ steps for every VM in S^* to have a price equal to 0, eliminating S^*. During the E^* (S^*) elimination process the "Ford and Fulkerson" algorithm (FindUnivAllocItems procedure) is called to identify the E^* (S^*) at most $\lceil \bar{p}/\Delta p \rceil$ times; therefore, since the "Ford and Fulkerson" algorithm (FindUnivAllocItems procedure) runs in polynomial time, the elimination process is also polynomial. □

However, applying both E^*-*increase* and S^*-*decrease* policies at the same iteration might trap the process in a cycle, *i.e.,* the set of excess supply and demand return to their previous state after a number of iterations as it is shown in [134]. Therefore, the E^*-*increase* (S^*-*decrease*) policy has to be deployed in isolation in each iteration until completely eliminating E^* (S^*) before changing to policy S^*-*decrease* (E^*-*increase*) targeting the set S^* (E^*). The convergence to a price vector

[4] Other Multi-item auctions like the ascending (descending) Vickrey-English (Vickrey-Dutch) auctions [134, 162] require to start their execution from the lowest (highest) possible price in the market for each item.

where both sets of excess supply and excess demand are empty, $E^* = S^* = \{\emptyset\}$, is guaranteed by the following monotonicity lemma proved in [30]:

Lemma 2: For any price vector $p > 0$, i) If $S^{*,k} = \{\emptyset\}$ and an E^*-*increase* price adjustment policy is applied at iteration k, then $S^{*,k+1} = \{\emptyset\}$; similarly, ii) if $E^{*,k} = \{\emptyset\}$ and an S^*-*decrease* price adjustment policy is applied at iteration k, then $E^{*,k+1} = \{\emptyset\}$.

A typical iteration of VED auction is presented in procedure "VED-Iteration" of Algorithm 1. Essentially, the procedure applies a E^*-*increase* policy (line 23) as long as E^* is not empty; otherwise, an S^*-*decrease* policy is applied (line 20) until set S^* is empty too and the VCG equilibrium has been found (lines 17-18).

4.3.2 Edge-MAP: Orchestrator Operation

The Edge-MAP Orchestrator component, is responsible for providing sufficient information to the INCPs participating in a market about profit opportunities. In that way, Edge-MAP aims to act beneficially for both AppSPs and INCPs by promoting the competition between INCPs, who can develop their own VM supply strategies over different cells/markets in their proximity. First of all, VMs offered by the same INCP are identical from the user perspective since her valuation is specific to the cloudlet location; therefore, she has no preference between two VMs that coexist at the same location. The following lemma associates the price of identical VMs with the set of VMs in excess demand.

Lemma 3: Identical VMs, in terms of users' valuation, can only co-exist in the set of excess demand, E^*, if their prices are equal.

Proof. Given the definition of demand correspondence, $D_m(p)$, in Eq. (4.2), the users show a preference among identical VMs with the same *valuation*, v, for the one with the smallest price, p^*, since it maximises their *net-valuation*, *i.e.,* $v - p^*$. Therefore, the only VM that could belong to the E^* set is the one with the smallest price. Therefore, if both items belong to E^*, their price is p^*. □

Next, consider an INCP managing a cloudlet that participates at a cell market during time-slot t with 10 VMs. If all of the VMs are in excess demand, then according to Lemma 3, the VMs have the same price p_h. Then Edge-MAP gives to the INCP the options of i) waiting for the E^*-*increase* policy to increase the p_h price by Δp and go to the next VED auction iteration, or ii) increasing the number of VMs participating in the market of the cell by at most $\Delta|h|$ additional VMs. We refer to the second option as E^*-*dimensioning* policy (Algorithm 1 line 6). In other words, via E^*-*dimensioning* identical VMs to the ones in excess demand are *supplied* into the cellular market in order to eliminate the excess demand set as we show next.

Lemma 4: The combined application of E^*-*dimensioning* and E^*-*increasing* policies in a single iteration of Algortihm 1, eliminates the excess demand set, E^*, in polynomial time.

Proof. Assume that $|\tilde{\mathscr{M}}|$ number of users request VMs from E^*, *i.e.,* $|\tilde{\mathscr{M}}| > E^*$. Then, if there are

available VMs to be offered by the INCPs, the E^*-*dimensioning* policy requires $|\tilde{\mathscr{M}}|-|E^*|$ steps, when $\Delta|h|=1$, or only 1 step, when $\Delta|h|=|\tilde{\mathscr{M}}|-|E^*|$, to eliminate the excess demand E^*, since the number of VMs in E^* will no longer be less than the number of users. However, in the worst case, the "Ford and Fulkerson" algorithm is called to identify set E^* in each iteration of Algortihm 1 twice, once for the E^*-*dimensioning* and once for the E^*-*increasing* policy. But, again, the identification of set E^* is bounded by $2\lceil \bar{p}/\Delta p \rceil$; that is, E^* elimination takes place in polynomial time. □

Data: p, $\mathscr{M}$, $\mathscr{H}$,Δp,$\Delta|h|$.

Result: p^{VCG},x,

Initialisation: $k=1, p^1=p$.
while *True* **do**
  Collect $D_m(p^k)\ \forall m \in \mathscr{M}$
  Estimate $E^{*,k}$,x^k.
  if $(E^{*,k} \neq \emptyset)$ **then**
    $\mathscr{H}'$:=E^*-dimensioning($E^{*,k}$, p^k,$\mathscr{H}$,$\Delta|h|$).
    $\mathscr{H}=\mathscr{H}\cup\mathscr{H}'$
  p^{k+1},x^{k+1}, $flag$:=VED-Iteration(p^k, $\mathscr{M}$,$\mathscr{H}$,Δp)
  if *(flag)* **then**
    return: p^{k+1},x^{k+1}
  k+:=1
end
VED-Iteration(p^k, $\mathscr{M}$,$\mathscr{H}$,Δp)
  Initialisation: Collect $D_m(p^{b,k})$, $\forall m \in \mathscr{M}$, Estimate $E^{*,k}$,x^k,$p^{k+1}=p^k$.
  if $(E^{*,k}==\emptyset)$ **then**
    Estimate $S^{*,k}$.
    if $(S^{*,k}==\emptyset)$ **then**
      return: p^{k+1},x^k, *True*.
    else
      $p_h^{k+1}=p_h^k-\Delta p,\ \forall h \in S^{*,k}$
    end
  else
    $p_h^{k+1}=p_h^k+\Delta p,\ \forall h \in E^{*,k}$
  end
  return: p^{k+1},x^k, *False*.

Algorithm 1: Edge-MAP on demand provisioning mechanism.

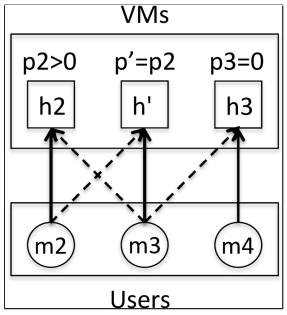


Figure 4.6: E^*-*dimensioning* example, VM h' is introduced at price $p' = p_2$.

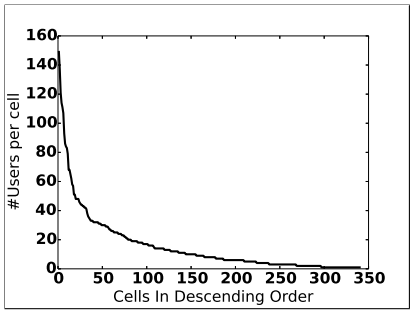


Figure 4.7: Users per cell in descending order, at 11am.

For example, in Fig. 4.5 where the set of excess demand is $E^* = \{h_2, h_3\}$, if the E^*-*dimensioning* policy introduces a VM h' that is identical to h_2, the excess demand set is immediately eliminated as we see in Fig. 4.6. Essentially, an INCP receives information about her VMs price and number in excess demand in a specific market. Then, by applying her own profit maximisation strategy, the INCP estimates a minimum price that she would accept for contributing additional VMs, denoted by p_{min}. If p_{min} is higher than the current market price, p_h, the INCP will wait for the E^*-*increase* policy to be applied and increase the p_h price. On the other hand, if $p_{min} < p_h$ she will supply this market with additional VMs. Nevertheless, elaborating on the INCPs' strategies is beyond the scope of this chapter. Note that Edge-MAP allows the application of the E^*-*dimensioning* policy for an INCP under the conditions that *a*) all of the INCPs' VMs are in excess demand, and *b*) the VMs to be included to the cell market are not currently involved in any other market.

Theorem 2. The Edge-MAP mechanism, as described in Algorithm 1, converges to the VCG equilibrium in polynomial time.

Proof. From Lemma 4, we have that the initially applied E^*-*dimensioning* and E^*-*increase* policies in Algorithm 1 eliminate E^* in polynomial time. After the E^* elimination, the S^*-*decrease* policy eliminates the excess supply S^* in polynomial time (Lemma 1) without affecting the already eliminated E^* set (Lemma 2). Thus, from Theorem 1, Algorithm 1 derives the VCG equilibrium after 2 sequential polynomial time processes; rendering Algorithm's 1 execution time polynomial. □

4.4 Edge-MAP Performance Evaluation

In this section, we demonstrate the performance of Edge-MAP. We begin by describing the setup of our evaluation before presenting our results.

4.4.1 Evaluation Setting

Mobility traces and cellular setting: The evaluation is based on a mobility dataset[5] developed in the context of the TAPASCologne project by using a state of the art mobility generator tool [183].

[5] http://kolntrace.project.citi-lab.fr

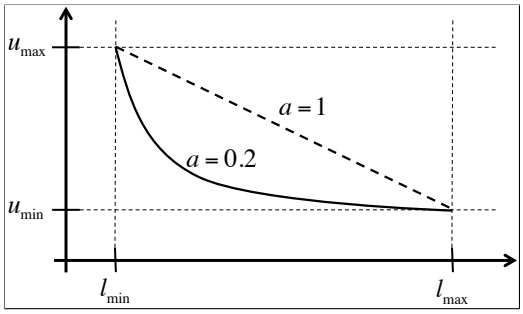


Figure 4.8: QoS function we consider for a LLA category.

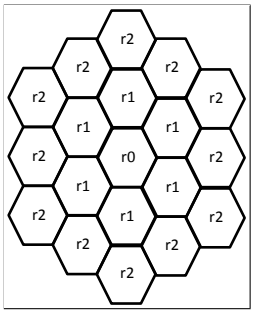


Figure 4.9: Allowed Market Participation of a cloudlet at r_0.

The dataset consists of 700,000 car journeys covering a region of 400 square kilometres over 24 hours. We create a cellular infrastructure by dividing the region into 864 hexagon cells of 1Km separation. We consider each vehicle as a mobile user, associated with a mobile device, and we focus on a single off-peak hour of the dataset, *i.e.,* 11am, which presents a good ratio of average vehicle speed and number of vehicles at each second, *i.e.,* $\sim$30 Km/h and $\sim$5200 vehicles, respectively. During this period, the population remains relatively constant, with approximately 13 vehicles leaving/ arriving every second. The number of vehicles/mobile users per cell for a snapshot of the data, at 11am, is shown in Fig. 4.7.

Application categories for LLAs: The QoS of the LLAs is naturally expressed as a decreasing function of the end user perceived round-trip time (RTT), in terms of latency x, to each cloudlet [62, 207]. Similarly to [99] for abstract resource allocation gains, we consider that each LLA is characterised by a decreasing QoS function of the general form:

$$u(x) = \left(\frac{u_{min}}{u_{max}} + \left(1 - \frac{u_{min}}{u_{max}}\right) \left(1 - \frac{x - l_{min}}{l_{max}}\right)^{\frac{1}{\alpha}} \right) \times u_{max} \tag{4.3}$$

The constants u_{max} (u_{min}) represents the maximum (minimum) QoS that the application user can achieve at the minimum (maximum) latency l_{min} (l_{max}), *i.e.,* $u(l_{min}) = u_{max}$ and $u(l_{max}) = u_{min}$. We set $u_{max} = 100$, $l_{min} = 4$ms[6], and $l_{max} = 500$ms assuming that all applications have identical latencies to their default cloud locations[7]. Moreover, function $u(\cdot)$ is convex for $0 < \alpha \leq 1$; that is, we set $\alpha = 0.2$ since LLAs' QoS is expected to be more sensitive to latency changes closer to l_{min}. We depict the QoS function for α values 0.2 and 1.0 in Fig. 4.8.

Based on Eq. 4.3, we create ten *LLA categories*, each associated to a QoS function that models a certain sensitivity of the LLA to network latencies. We use a different u_{min} value assigned $\{0, 10, 20, \ldots, 90\}$ for each category of LLA. In this way, different application categories have different gains from being provisioned at the edge, varying from 10, for $u_{min} = 90$ and less latency

[6]With recent advances in LTE technology, mobile operators reported handset-to-base-station latencies around 2 msec (RTT of 4 msec), see: http://news.itu.int/with-5g-looming-sk-telecom-reduces-lte-latency-to-just-2ms

[7]500ms is the maximum latency observed for Amazon Web Services according to CloudPing, available at http://www.cloudping.info.

Cells Distance Separation	1Km
Number of VMs per Cloudlet	20
Number of LLA categories	10
Δp, $\Delta\|h\|$	1
p_{min}	0
Time-slot duration	60s
Per hop in between cell latency	10ms
INCP market participation in cellular hops	1

Table 4.2: Edge-MAP Default Evaluation Setting

sensitive LLAs, to 100, for $u_{min} = 0$ and latency critical LLAs. We consider a number of ten LLA categories sufficient for the purpose of our evaluation since it is comparable to the currently considered types in the context of IoT [47, 119] and Tactile Internet [80].

Setting parameterisation: In the mobile environment we consider, we focus *only* on cloudlets located at the network edge, *i.e.,* base-station cells. In the default setting, cloudlets are allowed to advertise and offer their VMs to cellular markets that are up to 1 cell away, covering the cells denoted by r_0 and r_1 in Fig. 4.9 when the cloudlet is located at r_0. We set the inter-cell latency to 10ms and assume a tree-like backhaul topology [148], where the latency between cells increases linearly with the hop distance, *e.g.,* if a cloudlet is located 2 cells away from the cell a user is connected to, then the involved latency is 20ms. Given the network topology latencies, we round up the non-integer QoS values produced by Eq. 4.3, we parameterise the Edge-MAP mechanism by setting $\Delta p = 1$, $\Delta|h| = 1$, $p_{min} = 0$, while the time-slot duration equals 60s.

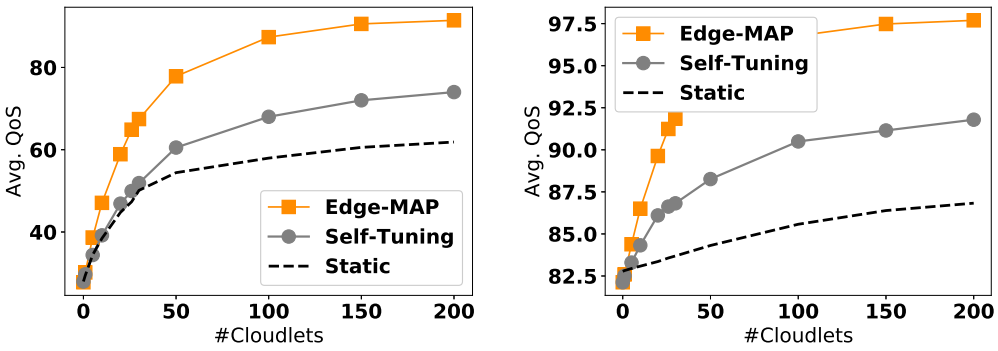
Considering the statistics provided by the Smart Insights' [14] survey regarding *i*) the user application engagement duration over a day, and *ii*) the smartphones market penetration percentage, we consider that 10% of the mobile users are engaged in a LLA at any second, *i.e.,* 520 users on average.

We set the capacity of each Cloudlet to 20 VMs, which is on average sufficient for serving even the most crowded cells, given the 10% users' participation we assume. Note that we limit each cell to host *at most a single cloudlet* in all the experiments. The default values of the experiment parameters are provided in Table 4.2.

LLA requests generation: We consider two approaches of generating service requests, namely:

- *Probabilistic requests*: Each user selects one of the 10 LLA categories according to a Zipf distribution that favours the most QoS sensitive application categories, *i.e.,* the most popular applications are the ones with the highest $u(l_{min})$. Similarly to edge caching systems [39], we set the Zipf's distribution exponent equal to 0.8.

- *Realistic requests*: Each user selects one of the 10 LLA categories based on the sequence of



(a) Probabilistic Requests

(b) Realistic Requests

Figure 4.10: Edge-MAP vs. Static, and Self-Tuning Provisioning

request arrivals in Google's cluster dataset[8]. In detail, we associate each LLA category to a "ParentID" field, that identifies the service, of the 10 most popular services in the dataset accounting for more than 200K requests. Then, by selecting random time intervals in the period of the seven hours that the dataset covers, users request LLA categories based on the sequence of "ParentID" fields that arrive into Google's cluster.

In both cases, we assume that users remain engaged to their requested LLA throughout their journey.

Cloudlets' deployment: We evaluate the impact of Edge-MAP in relation to the availability of cloudlet resources, which are incrementally deployed over the cells starting from the most crowded ones. In this way, we capture Edge-MAP's behaviour over the spectrum of different cloudlet infrastructure conditions; starting from under-deployed, where a single cloudlet exists only at the most crowded cell, to over-deployed, where there is an installed cloudlet at each cell. The turning point between over- and under- deployed infrastructure is taking place upon the deployment of the 26th cloudlet, where the available VMs, *i.e.,* 26×20=520 VMs, equal the average number of participating users, *i.e.,* 10% of the 5200 users in the dataset. Note that all cells support a marketplace no matter if a cloudlet is locally deployed. This implies that the required computing power for executing Edge-MAP's cellular component, at each cell, can be provided by the current base stations. Furthermore, we assume that Edge-MAP's orchestrator component presents a stable feedback communication performance equal to the latency imposed by the cellular distance between a market and a cloudlet's location. Clearly, throughout our evaluation we consider Edge-MAP's markets deployment as the default applied framework by telecom providers for serving their mobile users' LLA requests.

4.4.2 Simulation Results

The results presented next have been averaged over 100 executions.

Impact of on-demand provisioning: We compare Edge-MAP against *i*) Static and *ii*) Self-

[8] Available at https://research.googleblog.com/2010/01/google-cluster-data.html.

Tuning [117] provisioning in terms of average QoS. In static provisioning, we assume that each AppSP is aware of his aggregate service demand and allocates the portion of VMs at each cloudlet that corresponds to this demand. For example, given the cloudlet capacity of 20 VMs and an LLA $s \in \mathscr{S}$ accounting for half of the generated requests, static provisioning will allocate statically 10 instances of s at each cloudlet for the entire duration of the simulation.

On the other hand, in self-tuning, the provisioning of LLAs takes place periodically, *i.e.,* repeating at regular time-slots, at each cell. The provisioning of LLAs in the self-tuning approach at a time-slot t is based on *i*) the demand of each LLA observed during the time-slot $t-1$ and *ii*) the QoS gain of LLAs from being provisioned at the edge. For instance, if LLA s is requested on average by, say seven users, during time-slot t at a specific cell, the AppSP will bid for up to seven VMs at the respective cellular market at time-slot $t+1$. If LLA s is the service with the highest QoS gain, then seven VMs will be allocated to it at the price of the next seven offered prices, *i.e.,* generalised second price auction. Note that the original self-tuning approach proposed in [117] is a generalised second price combinatorial auction requiring offline execution. In order to create an online distributed variation of the self-tuning approach that it is comparable to Edge-MAP, we limit cellular markets in offering VMs that are located only at the current cell. That is, we eliminate the combinatorial difficulty of the problem by offering *identical* VMs in terms of QoS gain, since the gain of each AppSP increases linearly with the number of allocated VMs until reaching its expected demand.

Edge-MAP outperforms the other approaches in terms of average QoS in both probabilistic and realistic request generation settings, while the self-tuning approach is superior only to the static provisioning approach, as we see in Fig. 4.10. Clearly, the average QoS is lower for the Zipf-based probabilistic request generation due to the correlation between each service's popularity and QoS gain, *i.e.,* the most demanding service is the most popular, and lack of resources results in the faster deterioration of the system average QoS. In the remaining experiments we present the results of the probabilistic request generation which is the more challenging case.

Impact of time-slot duration: In Fig. 4.11 we depict the average QoS under different time-slot durations, namely 10, 60, and 120 seconds, capturing an increasing configuration time overhead related to the VM management and potential applications state migration. As expected, a longer time-slot duration leads to QoS deterioration due to the decrease in Edge-MAP provisioning responsiveness to LLA demand changes, caused by mobile users' *i*) handovers and *ii*) arrivals/departures to/from the system.

Benefit of per-cell markets: Fig. 4.12 demonstrates the increase in the execution time of the mechanism when the provisioning of resources takes place via a fixed number of markets instead of deploying one marketplace at each cell. In particular, we consider a centralised scenario, *i.e.,* a single market in the system, a scenario with two markets, *i.e.,* each market is responsible for roughly half of the cells in the system, and three markets, *i.e.,* each market is in charge of the one thirds of the cells in

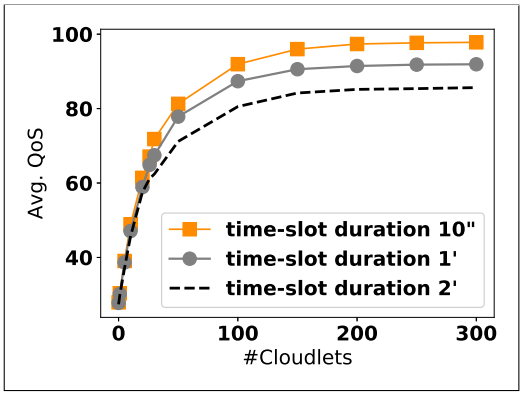


Figure 4.11: Time-slot Duration QoS Impact

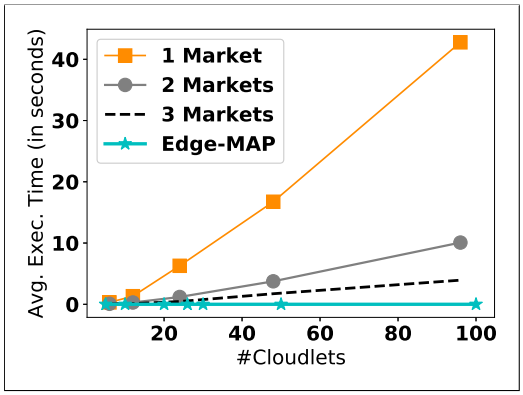
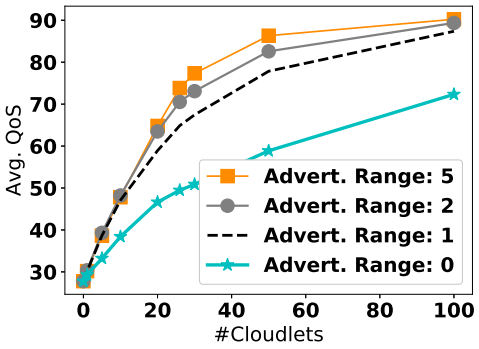
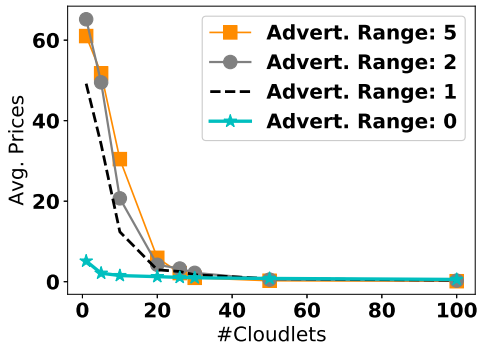


Figure 4.12: Centralised Markets: Execution Time Impact.



(a) Avg. QoS.

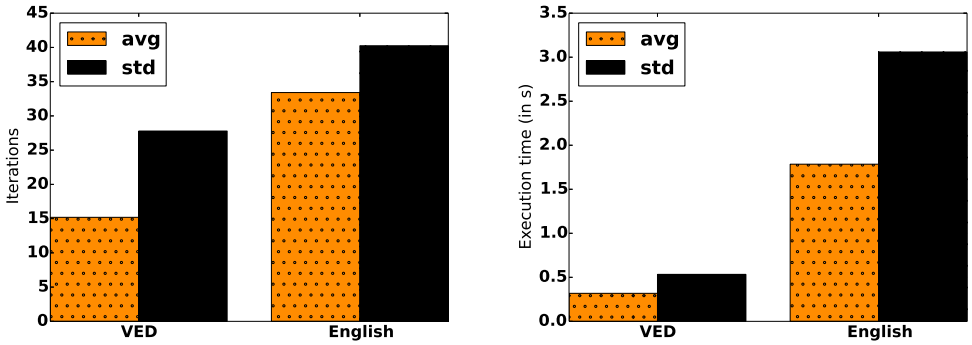


(b) Avg. Price.

Figure 4.13: Local vs. PooI of Virtual Resources

the system. The reason behind Edge-MAP's negligible execution time compared to the fixed number of markets is that the cellular based markets involve a considerably smaller number of bidders, since they include only the users that are connected to the current base station, as well as the number of VMs. Note that Edge-MAP scales gracefully as the number of cloudlets in the system increases while, in the case of fixed markets, the time increases linearly due to the polynomial execution time nature of VED auctions. The execution times are computed using a 2.2 GHz Intel Core i5 processor.

Benefit of pooling of interconnected virtualised resources: Fig. 4.13a depicts the QoS gain from allowing cloudlets to offer their VMs to distant markets, defined by the cellular advertisement range with respect to the relative position of a cloudlet location, *i.e.,* each cloudlet is placed on ring 0 (r_0 in Fig. 4.9). Undoubtedly, there is a higher QoS gain when INCPs act as a pool of interconnected virtualised resources that can be offered over different cells, *i.e.,* rings range 2 (r_0, r_1, r_2 in Fig. 4.9) and 5, than offering their resources to the *local* cellular market where they are placed, *i.e.,* rings range 0. The reason is that idle VMs have the opportunity of being utilised by users connected to a different



(a) Iterations Comparison.

(b) Execution Time Comparison.

Figure 4.14: VED vs. English Auctions

cellular market, where due to the *limited elasticity* of the local cloudlet there are no available VMs for serving their requests. Furthermore, from Lemma 3, we know that the VMs offered by a single cloudlet to a single market can be allocated only if they have identical prices. In other words, the price that a INCP can get from his resources is *market specific*. Therefore, offering VMs to distant markets is beneficial for the income of the INCPs since they have the opportunity to diversify their prices. This is clear in Fig. 4.13b, where over the under-deployed zone, *i.e.,* number of cloudlets 1 to 26, the average price of VMs is substantially higher than the case when VMs are only offered locally, rings range 0. On the other hand, at the over-deployed zone, *i.e.,* number of cloudlets 26 onwards, the average price is approaching the value of zero, due to the abundance of resources and the competition conditions that are created.

Benefit of VED auctions: Lastly, in order to demonstrate Edge-MAP's scalability in increasing workloads, we consider the extreme case where all mobile users request a LLA in a setting where 26 cloudlets are deployed, each one capable of supporting 200VMs. Figures 4.14a and 4.14b present the comparison of Edge-MAP against an Edge-MAP's variation that relies on Vickrey-English (VE) ascending auctions instead of VED. Clearly, VED auctions dominate over the VE ones both in terms of iterations and execution time, since VED requires less than a second to derive the new VCG equilibrium as opposed to VE whose execution time might exceed the 4 seconds. Therefore, VED auctions are ideal for repetitive allocation settings where they can take advantage of the previously found equilibrium for decreasing their execution time.

4.5 Conclusions

In this chapter, we investigated the deployment of LLAs in a setting of mobile connectivity. Along these lines we presented Edge-MAP, a polynomial time mechanism tailored to the on demand provisioning of LLAs for mobile users. At a micro-level, Edge-MAP operates on cellular based markets using VED auctions to perform robust resource allocation. Edge-MAP on macro-level fosters the

competition among INCPs by providing feedback with respect to profit opportunities on different markets. Our evaluation verified Edge-MAP's design capability of taking into account the inherent challenges of the LLA provisioning market we consider.

Chapter 5

Enabling LLAs in Fixed Connectivity Settings

5.1 Introduction

In this chapter, we investigate the emerging market of provisioning heavily stateful LLAs over the edge/fog infrastructure that consists of independent cloudlets. In our context, heavily stateful LLAs cannot be migrated to another cloudlet location due to the size of their runtime data, generated by their users, within a reasonable timeframe. In other words, we assume that heavily stateful LLAs cannot be suspended for serving another request without a user quitting her session. In the setting we consider, computation is available either at the edge or at the middle-tier locations of the network, in the form of cloudlets, and/or at distant clouds/data centres (Fig. 5.1). We argue that service provisioning over cloudlets is expected to take place in a decentralised and uncoordinated environment. Given that cloudlet resources are limited, the key challenge is to create a market that operates on a per-request basis for offering the finest possible resource allocation granularity. We aim to provide answers to the fundamental questions of: *i*) *how should the cloudlet resources be allocated over time to different applications/services?*, and *ii*) *how much should a cloudlet charge an application,* i.e., *the application producer/creator?*, for the case of heavily stateful LLAs. Here, we present a decentralised pricing mechanism that answers both questions, while addressing the challenges of dynamic service provisioning over the fog computing infrastructure. Note that, in the case of not heavily stateful LLAs, the Edge-MAP mechanism, described in Chapter 4, could be deployed since mobile connectivity conditions are considered more generic compared to the fixed ones.

Our starting point is the spot pricing mechanism [21], which creates an auction-based market for available cloud computing resources. Cloud providers determine their spot price at regular time intervals subject to their resources' demand. Then, at each time-interval, the users' requests that bid above the spot price are accepted while the rest of the users are suspended until the spot price falls below their bid. Clearly, the spot pricing distributed solution to the problem of allocating

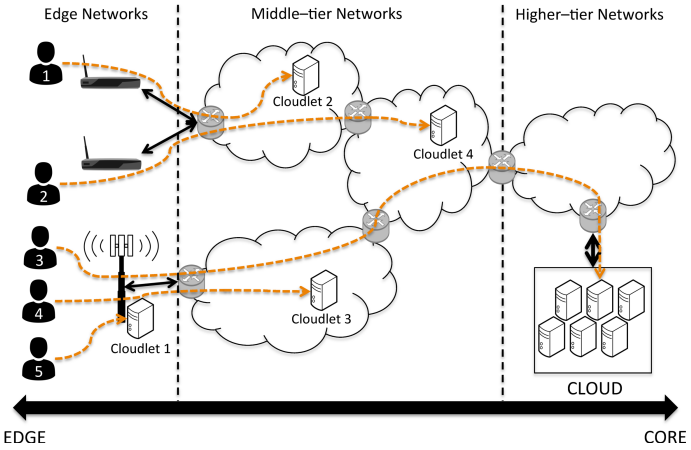


Figure 5.1: End-users on-path application request service by cloudlets located at the edge and middle-tier locations of the network.

cloud resources is suitable for tasks that can be disrupted; hence, spot pricing is unsuitable for LLA provisioning since interruption affects their QoS.

For this reason, we introduce FogSpot, a pricing mechanism that associates each cloudlet with a spot price for on-path, on-demand, distributed LLA provisioning. In FogSpot, cloudlets offer their resources in the form of Virtual Machines (VMs) via collocated markets. As LLA requests are forwarded towards a default execution location, they interact with on-path cloudlet markets. If the spot price of a market is below the gain the LLA will have if it is served, an available VM is allocated to serve the request; otherwise, the request is rejected and continues its journey towards the cloud. For example, in Fig. 5.1, the request of user 2 at first interacts with cloudlet 2 which rejects it; the request then continues its journey to cloudlet 4, that finally accepts it. On the other hand, the request of user 3 fails to get served by all cloudlets 1, 3 and 4, reaching its final execution location at the cloud. FogSpot addresses explicitly the heavily stateful requirement of seamless users' engagement to LLA instances while setting the spot price of each cloudlet with the aim of maximising either its revenue or utilisation. To this end, the main technical contributions of this chapter are as follows:

1. We argue about the need for an on-path, on-request provisioning mechanism in the case of heavily stateful LLAs.

2. We introduce FogSpot, the first spot pricing mechanism for edge/fog computing resources, that guarantees end users' requests truthfulness in the case of LLAs' provisioning while focusing in maximising either each cloudlet's revenue or resource utilisation.

3. We illustrate the merits of FogSpot via extensive simulations in simple topologies.

5.2 Design Rationale

The challenge here is the design of a market mechanism tailored to the provisioning of LLAs over an uncoordinated cloudlet infrastructure. Our design has to address explicitly the corresponding provisioning challenges of *i*) the cloudlets' resource discovery, and *ii*) the cloudlets' limited elasticity.

In our context, cloudlet resource discovery is defined as the process of finding appropriate computing resources for provisioning LLAs. Resource discovery over a set of geographically distributed clouds is a challenging task [74] that is closely associated with the process of resource monitoring [18], for taking full advantage of the cloud's capability of allocating and releasing resources on-demand [209]. In the environment we envision, cloudlets points of presence are expected to exceed by far the number of clouds. Therefore, discovering and allocating resources, as a response to a continued monitoring process, would face scalability issues.

Given that LLA requests are forwarded in the network towards a distant Data Centre, we argue for both *on-path and on-demand application provisioning*. In particular, by applying *on-path provisioning*, there is no need for a centralised resource discovery process, since resources are discovered in real time. Furthermore, applications are not required to monitor the usage of their resources since the *on-demand provisioning* guarantees their full utilisation, *i.e.,* each instance at any time point is utilised by a user. We argue that these design choices enable the most promising and incrementally deployable conditions for the problem of LLA provisioning we tackle, providing an uncoordinated and distributed solution similar to the content delivery via *on-path* caching [150].

In this chapter, we argue *cloudlet pricing schemes should follow a pay-as-you-go structure in terms of application user engagement duration* in order to promote the on-demand provisioning of applications. In other words, the LLA providers should be charged based on the time their users occupy a cloudlet instance (*e.g.,* π dollars per 1 ms).

5.3 System Model

We consider a set $\mathscr{S} \triangleq \{1,2,...,S\}$ of LLAs and a set $\mathscr{D} \triangleq \{1,2,...,D\}$ of cloudlets. The engagement duration of an LLA $s \in \mathscr{S}$ has an exponential distribution with rate parameter μ_s, where $1/\mu_s$ is its mean engagement time. We assume that each LLA can be provisioned at a distant cloud/data centre whose capacity is sufficient, in all cases, for serving the total number of LLA requests it receives. In particular, each LLA request is forwarded via a *path* from the access point of an end user to a default application cloud/data centre, $\tilde{D}_s \ \forall s \in \mathscr{S}$. Each path p is considered as a distinct traffic class traversing a set of $\mathscr{D}_p$ cloudlets. Specifically, a class p request of LLA s experiences a QoS improvement $u^d_{s,p}$, in terms of network conditions, for each second that the end user remains engaged to an LLA instance at cloudlet $d \in \mathscr{D}_p$, as opposed to the default Data Centre $\tilde{D}_s$. We denote that set of request classes in the network by $\mathscr{P} \triangleq \{1,2,...,P\}$. Note that a cloudlet d receives traffic from a set of classes, $\mathscr{P}_d$; that is, the gain of a request of LLA s served at d is class-specific. In our system, the LLA requests of each class are generated according to a Poisson process of rate $\lambda_{s,p}$ requests per

System Model	
$\mathscr{S}$	Set of LLAs.
$\mathscr{D}$,$\tilde{D}_s$	Set of cloudlets, default data centre of LLA s.
μ_s	Exponential distribution engagement duration rate of application s.
$\mathscr{P}$	Set of requests' classes.
$\mathscr{D}_p$	Set of cloudlets serving class p.
$\mathscr{P}_d$	Set of requests' classes arriving at cloudlet d.
$\lambda_{s,p}$	Poisson process arrival rate of LLA s class's p requests.
$u^d_{s,p}$	Per second QoS gain of LLA s at cloudlet d for class p, in terms of network condition.
$J_s(\cdot)$	Average aggregated utility of LLA s .
$y^d_{s,p}$,y_s	Request provisioning rate of LLA s class p at d, provisioning rate matrix of LLA s.
C_d	Number of VMs at cloudlet d.
ρ_{thres}	Target utilisation level of resources.
FogSpot On-Demand Provisioning	
π_d	Spot price of cloudlet d.
FogSpot Spot Price Derivation Mechanism	
$\mathscr{P}_{d,s}(\pi^t_d)$	Set of classes of LLA s with gain in being provisioned at price π^t_d at cloudlet d.
$X^d_s(t)$	Arrival rate of LLA s at d during iteration t that can be provisioned.
$Y^d_s(t)$	Admitted request rate of LLA s by cloudlet$d$ during iteration t.

Table 5.1: FogSpot Notation

second $\forall s \in \mathscr{S}$ and $\forall p \in \mathscr{P}$.

In our setting each request concerns the allocation of a single VM. Then, let the requests of LLA s of class p be admitted at cloudlet d according to a rate $y^d_{s,p}$; we define the *provisioning rate matrix* of LLA s, $y_s \triangleq (y^d_{s,p} : p \in \mathscr{P}, d \in \mathscr{D})$. Note that if $d \notin \mathscr{D}_p$ then $y^d_{s,p} = 0$ for all $s \in \mathscr{S}$. The admitted request rates of a class respect the generation Poisson process rate, *i.e.,* class p cannot admit more than the generated requests:

$$\sum_{d \in \mathscr{P}_d} y^d_{s,p} \leq \lambda_{s,p}. \tag{5.1}$$

Note that constraint (5.1) implies that the default execution location of LLA s, $\tilde{D}_s$, admits requests according to a Poisson process of rate $\lambda_{s,p} - \sum_{d \in \mathscr{P}_d} y^d_{s,p}$, if the admission rates $y^d_{s,p}$ also characterise a Poisson process. We use $J_s(y_s)$, $\forall s \in \mathscr{S}$, to denote the utility of LLA s as a function of y_s in terms of aggregated per second QoS gain expressed as:

$$J_s(y_s) = \frac{1}{\mu_s} \sum_{p \in \mathscr{P}} \sum_{d \in \mathscr{D}_p} u^d_{s,p} y^d_{s,p}, \tag{5.2}$$

where $y^d_{s,p}/\mu_s$, is the average number of active class p request s sessions at cloudlet $d \in \mathscr{D}_p$, *i.e.,* the average number of allocated VMs to class p of LLA s, by Little's Law.

Let C_d be the number of identical VMs, in terms of dedicated CPU, memory, and storage resources, that cloudlet $d \in \mathscr{D}$ can support.[1] Observe that the average utilised VMs have to respect

[1]The presented mechanism could be applied over different types of instances by referring immediately to the cloudlet resources, that are allocated in different amounts for different types of VMs.

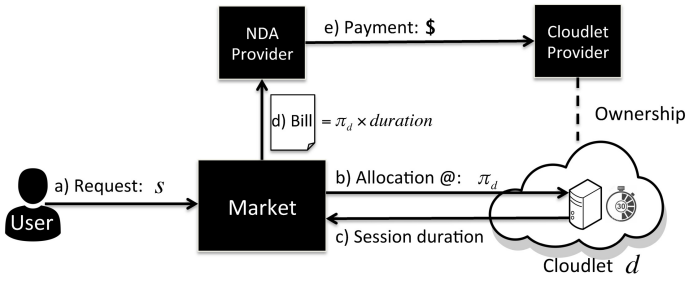


Figure 5.2: FogSpot On-Demand Provisioning Overview

the VM limit restriction:

$$\sum_{s\in\mathscr{S}} \frac{1}{\mu_s} \sum_{p\in\mathscr{P}_d} y_{s,p}^d \leq \rho_{\text{thres}} C_d,\ \forall d \in \mathscr{D}, \tag{5.3}$$

where ρ_{thres} is the target utilisation level of the resources taking values in $[0,1]$.

5.4 FogSpot Mechanism

5.4.1 FogSpot: On-Demand Provisioning

An overview of the FogSpot on-demand provisioning scheme is depicted in Fig. 5.2, where we consider an end user request of LLA $s \in \mathscr{S}$ that arrives at the market of cloudlet $d \in \mathscr{D}$ (step a). The interaction of the request with each market is characterised by the following properties:

A1) The request is for a single VM.

A2) The request is associated with a bid, $b_{s,p}$, that expresses its user's class p willingness to pay for a VM for each engagement time-unit of the user to an LLA s instance.

A3) The request is not queued at the market, *i.e.,* the request is either served or rejected immediately.

A4) The VM allocation time overhead has no impact on LLA's QoS.

In more detail, let the spot price at cloudlet d be π_d. Upon the arrival of a request for LLA s, that is associated to a bid $b_{s,p}$. The market operates according to the following rules:

R1 If there are not available VMs, reject the request.

R2 Else, if there are available VMs:

- If $b_{s,p} \geq \pi_d$, a VM of cloudlet d is allocated to serve the request at price π_d for each time unit of user engagement.
- Else if $b_{s,p} < \pi_d$, the request is rejected.

Assuming $b_{s,p} \geq \pi_d$ and the existence of available VMs, the allocation is taking place and the cloudlet starts a timer for keeping track of user's engagement duration (step b). After the session completion/application termination, the cloudlet informs the market about the engagement duration

(step c). Then, the market verifies the duration and converts it to a bill that equals the engagement duration times the agreed per time unit service price π_d (step d). Finally, the bill is sent to the corresponding LLA provider which eventually pays the cloudlet provider for its service.[2]

Next, we explain how bids $b_{s,p}$ for users' LLA requests are derived $\forall s \in \mathscr{S}$ and $\forall p \in \mathscr{P}$. Let $u^d_{s,p}$ be the per time unit QoS gain of class p LLA s when served at cloudlet d, instead of its default cloud. Bidding truthfulness is defined in the following straightforward way:

Definition 1. *A bid $b_{s,p}$ for class p LLA s provisioning at cloudlet d is truthful iff $b_{s,p} := u^d_{s,p}$, where $u^d_{s,p}$ is the per time unit QoS gain of the requested LLA at d.*

In other words, a bid is truthful when it equals the actual gain of the served application. Then, given a spot price for this market, *i.e.* π_d, the LLA s utility rate from using the cloudlet d is defined follows:

Definition 2. *The utility rate of class p LLA s from bid $b_{s,p}$ and QoS gain $u^d_{s,p}$ is:*

$$\text{utility rate} = \begin{cases} u^d_{s,p} - \pi_d, & \text{if } b_{s,p} \geq \pi_d, \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 1. *Under rules R1-R2, the request bids are truthful.*

Proof. Let $b_{s,p}$ be the price of a buy order for a class p application s with $u^d_{s,p}$. We investigate the following possible cases:

- If $u^d_{s,p} \geq \pi_d$ then the bidder would win the item with a truthful bid as well as an overbid.
- If $b_{s,p} > \pi_d > u^d_{s,p}$, then overbidding would return a negative utility rate, as opposed to a truthful bid for which the utility rate is 0.
- If $\pi_d > u^d_{s,p}$ then the bidder would loose the item with a truthful bid as well as an underbid.
- If $u^d_{s,p} \geq \pi_d > b_{s,p}$ underbidding returns a utility rate equal to 0, as opposed to a truthful bid for which the utility rate is non-negative.

From the previous cases it is clear that truthful bidding is the dominant strategy. □

Since the buy orders must be truthful for a rational user, the strategy of their deployment is straightforward, *i.e.,* assigning a price $b_{s,p} = u^d_{s,p}$ to each request. Therefore, a request has to simply be associated to its involved QoS per time unit gain at a given cloudlet. This result contributes to the real time application provisioning since the end users do not have to know their per time unit gain with respect to each cloudlet in the network. Next, we focus on the derivation of cloudlets' spot prices.

[2]The technical details of the explained process, related to the security, verification, etc., are beyond the scope of this Chapter although orthogonal to its contribution.

5.4.2 FogSpot: Spot Price Derivation Mechanism

Next, we describe the FogSpot derivation of spot price for a given cloudlet. We start by analysing the queuing model we consider at each cloudlet, given the exponential arrival request rates and service times, before proceeding to the actual spot price derivation steps of FogSpot.

FogSpot Cloudlet Queueing Model

FogSpot on-demand provisioning treats each cloudlet d as a First Come Fist Served (FCFS) Queueing system of C_d identical parallel servers with zero size queues, since the requests are not queued. A cloudlet receives requests at exponential interarrival times with rate $\sum_{s\in\mathscr{S}}\sum_{p\in\mathscr{P}_d} y^d_{s,p}$ while the user engagement duration, *i.e.,* the service of a request, has application specific exponential interarrival times of rate $\mu_s\ \forall s\in\mathscr{S}$. In other words, each cloudlet d is a M/M/C_d/C_d/FCFS system, dictating the admitted provisioning rate for LLA s at d $Y^d_s=\sum_{p\in\mathscr{P}_d} y^d_{s,p}$ as we explain next.

Let X^d_s be the arrival rate of LLA s request that arrive at cloudlet d. The effective rate at which LLA s can be provisioned, Y^d_s, equals:

$$\begin{aligned} Y^d_s &= X^d_s \times P_d(\text{Cloudlet is not fully occupied}) \\ &= X^d_s \times \Big(1 - P_d(\text{Cloudlet is fully occupied})\Big) \\ &= X^d_s \times \Big(1 - P_d(n=C_d)\Big). \end{aligned} \tag{5.4}$$

Clearly, requests of LLA s are rejected from cloudlet d at rate $X^d_s - Y^d_s$ while the steady state probability of having all VMs occupied, $P_d(n=C_d)$, is:

$$\begin{aligned} P_d(n=C_d) &= \frac{\rho^{C_d}}{C_d!}P_d(n=0) \\ &= \frac{\rho^{C_d}}{C_d!}\left[\sum_{n=0}^{C_d}\frac{\rho^n}{n!}\right]^{-1}, \end{aligned} \tag{5.5}$$

where parameter $\rho=\sum_{s\in\mathscr{S}} X^d_s/\mu_s$ as described in [115]. Note that rate Y^d_s characterises a Poisson process. Lastly, it can be shown that as we increase the number of VMs capacity at d, C_d, the $Y^d_s \to X^d_s$.

Spot Price Derivation Process

FogSpot relies on Dutch auctions for deriving the spot price of the instance. Next, we describe the steps of the spot price derivation for a single cloudlet d, π_d, assuming that the rest cloudlets in the network have already derived their own spot price. To begin with, we initialise spot price π^t_d with a very high value $\pi_{\max}$, *i.e.,* a few times higher than any reasonable market price, so that $Y^d_s(t)=0$ $\forall s\in\mathscr{S}$ at $t=0$. Then, the steps followed are:

Step 1: Each LLA s estimates the traffic that could be served for price π^t_d, $X^d_s(t)$. At first, it identifies the classes whose gain after the price deduction is positive when served at d, $\mathscr{P}_{d,s}(\pi^t_d)\subseteq\mathscr{P}_d$, defined as:

$$\mathscr{P}_{d,s}(\pi^t_d)=\Big\{p : d\in\mathscr{D}_p \text{ and } u^d_{s,p}>\pi^t_d\Big\}. \tag{5.6}$$

Then:

$$X_s^d(t) = \sum_{p \in \mathscr{P}_{d,s}(\pi_d^t)} \Big(\lambda_{s,p} - \sum_{d' \in \mathscr{D}_p : u_{s,p}^{d'} > u_{s,p}^d} y_{s,p}^{d'} \Big), \tag{5.7}$$

where $\sum_{d' \in \mathscr{D}_p : u_{s,p}^{d'} > u_{s,p}^d} y_{s,p}^{d'}$ is the request provisioning rate of LLA s over the cloudlets with a higher QoS with respect to class p, *i.e.,* the cloudlets that are deployed between the end users and cloudlet d along a path that defines class p and therefore are *closer* to the end users. The LLA s then submits value $X_s^d(t)$ to cloudlet d. Note that each cloudlet d does not require information related to the requests' classes that serves for each LLA , since it is interested in the aggregated rate of each LLA s, $X_s^d(t)$.

Step 2: Cloudlet d estimates the provisioning request rate of LLA s, $Y_s^d(t)$, from (5.4). Note that the information about the cloudlet's capacity in terms of VMs does not have to be shared with the LLA providers, who are only interested in the request provisioning rate of their service at d.

Step 3: Cloudlet d applies one of the following termination criteria:

Revenue maximisation (RevM): The Dutch auction terminates if $\pi_d^t = 0$. Then, the spot price is set based on the candidate price that maximises the revenue of cloudlet d :

$$t^* = \arg\max_t \Big(\pi_d^t \times \sum_{s \in \mathscr{S}} Y_s^d(t) \frac{1}{\mu_s} \Big). \tag{5.8}$$

Then $\pi_d = \pi_d^{t^*}$ and $Y_s^d = Y_s^d(t^*)$ for each $s \in \mathscr{S}$.

Social welfare maximisation (SWM): The Dutch auction terminates either when the provisioning requests cover cloudlet's capacity, *i.e.,* $|\sum_{s \in \mathscr{S}} Y_s^d(t)/\mu_s - \rho_{\text{thres}} C_d| \leq \varepsilon$, or when $\pi_d^t = 0$. In both cases SWM policy serves the highest possible volume of LLA requests. The spot price is set to $\pi_d = \pi_d^t$ while the accepted request rates are set to $Y_s^d = Y_s^d(t)$ for each $s \in \mathscr{S}$. [3]

Step 4: Cloudlet d decreases spot price by $\Delta\pi$ according to the Dutch auctions, *i.e.,* $\pi_d^{t+1} = \pi_d^t - \Delta\pi$, and proceeds time $t = t + 1$.[4]

Addressing Truthfulness Concerns

Assume that LLA s is untruthful in the provisioning request rate that declares at cloudlet d, $\tilde{X}_s^d$ instead of X_s^d. Then, if $\tilde{X}_s^d > X_s^d$ cloudlet d will eventually detect this false declaration by monitoring the under-utilisation of its resources by LLA s. Similarly, if LLA s declares a $\tilde{X}_s^d < X_s^d$, the derived spot price at d will be reduced from the actual price π_d to $\tilde{\pi}_d$, *i.e.,* $\tilde{\pi}_d < \pi_d$, which means that the actual admitted requests will be higher than the expected ones. Then, cloudlet d will again be able to detect the untruthful bidding by monitoring the over-utilisation of its resources by LLA s. Especially when cloudlet d applies a SWM approach, a price $\tilde{\pi}_d < \pi_d$ increases the competition for resources and LLA s will have less chances in finding available VMs at d since the cloudlet utilises its resources completely.

[3] Note that, ρ_{thresh} serves the purpose of achieving a spot price higher than 0, since from Eq. 5.4 we see that the utilisation of the system can never be 100%.

[4] $\Delta\pi$ can be chosen in a way that $\pi_d^t \geq 0$.

Spot Price Derivation Frequency

The spot price derivation process is designed to take place in an uncoordinated way. A centralised approach would be practically infeasible due to the synchronisation overhead among the numerous cloudlets' Points-of-Presence. That is, we assume that in FogSpot each cloudlet triggers the spot price derivation mechanism periodically, *i.e.,* every Δt_d seconds.

5.5 FogSpot Game Theoretic Analysis

In this section, we analyse the uncoordinated price derivation performance of FogSpot mechanism from a game theoretic perspective.

5.5.1 Cloudlet Spot Pricing as a Game

We consider each cloudlet d as a player whose action is related to the spot price selection over a set of prices $\Pi_d = (u^d_{s,p} : \forall s \in \mathscr{S}, \forall p \in \mathscr{P}_d)$. We associate each cloudlet d to a payoff function $\mathscr{R}_d(\cdot)$ mapping $\Pi_1 \times \Pi_2 \times ... \times \Pi_D$ to a real value, where the elements of $\Pi_1 \times \Pi_2 \times ... \times \Pi_D$ are referred as *action combinations* or *states*. That is, *pure Nash equilibrium* is defined as a state $\pi = (\pi_1, ..., \pi_D)$ such that, for each cloudlet d, there is no action that can increase its payoff, *i.e.,* $\mathscr{R}_d(\pi_1, .., \pi_d, ..., \pi_D) \geq \mathscr{R}_d(\pi_1, .., \pi'_d, ..., \pi_D)$ for any $\pi'_d \in \Pi_d$.

Consider a directed graph $\mathscr{G} = (\Pi, E)$ which nodes consist of the states in $\Pi = \Pi_1 \times \Pi_2 \times ... \times \Pi_D$ space and edges E connect only states that differ by a single component causing an improvement to the corresponding player's payoff function, *i.e.,* there is an edge (π, π') iff states $\pi = (\pi_1, .., \pi_d, ..., \pi_D)$ and $\pi' = (\pi_1, .., \pi'_d, ..., \pi_D)$ differ only in component π'_d and $\mathscr{R}_d(\pi') > \mathscr{R}_d(\pi)$. From [157], we know that if the graph is acyclic then the *Nash dynamics converge* to a pure Nash equilibrium (the sinks of the graph), leading us to the proposition presented in [76] :

Proposition 2. *If the Nash dynamics converges, then there is a pure Nash equilibrium.*

Next, we investigate the settings under which FogSpot derives a pure Nash equilibrium to the game of cloudlet pricing for stable demand conditions, *i.e.,* $\lambda_{s,p}$ remains stable for each $p \in \mathscr{P}$ and $s \in \mathscr{S}$. Nash equilibrium indicates that FogSpot derives a stable solution in an uncoordinated way which also assures the maximum possible spot price at each cloudlet.

5.5.2 FogSpot Nash Equilibrium Derivation of SWM Policy

Let $Y^d_s(\pi)$ be the provisioning rate of service s at cloudlet d when the state of spot prices is π. The payoff function of SWM policy in FogSpot is:

$$\mathscr{R}^{\text{SWM}}_d(\pi) = \begin{cases} 1.0, & \text{if } |\sum_{s \in \mathscr{S}} Y^d_s(\pi)/\mu_s - \rho_{\text{thres}} C_d| \leq \varepsilon \\ & \text{or } \pi_d = 0, \\ 0.0, & \text{otherwise.} \end{cases} \tag{5.9}$$

Note that as π_d decreases, $\sum_{s \in \mathscr{S}} Y^d_s(\pi)/\mu_s$ increases.

Theorem 1. *The FogSpot mechanism converges to a pure Nash equilibrium with the highest possible payoff in the spot pricing game, when the SWM policy is applied.*

Proof. Let the current state of spot prices be $\pi = (\pi_1, ..., \pi_D)$ and cloudlet d is the next that triggers the spot price derivation mechanism. Then we know that the new spot price of cloudlet d, π'_d, which is derived by the SWM policy, will be either *i)* $\pi'_d < \pi_d$ such that $\mathscr{R}_d^{\text{SWM}}(\pi') > \mathscr{R}_d^{\text{SWM}}(\pi)$ in case that $|\sum_{s\in\mathscr{S}} Y_s^d(\pi)/\mu_s - \rho_{\text{thres}} C_d| > \varepsilon$, or otherwise *ii)* $\pi'_d = \pi_d$ and therefore state $\pi' = \pi$.

Clearly, FogSpot uncoordinated execution of SWM policy forms a directed graph whose nodes cover the states existing in space Π. SWM policy starts from state $\pi_{\text{init}} = (\pi_1 = \pi_{\max}, ..., \pi_D = \pi_{\max})$, *i.e.,* $Y_s^d(\pi_{\text{init}}) = 0$ for all $d \in \mathscr{D}$ and $s \in \mathscr{S}$. After that, the execution of the mechanism by a cloudlet d leads to a new state $\pi = (\pi_1 = \pi_{\max}, ..., \pi_d, ..., \pi_D = \pi_{\max})$ where $\pi_d < \pi_{\max}$ and $\mathscr{R}_d^{\text{SWM}}(\pi) > \mathscr{R}_d^{\text{SWM}}(\pi_{\text{init}})$. In general, edges in SWM policy graph connect only states that differ by a single price that is always decreased, causing an improvement to the payoff function of the corresponding cloudlet. Clearly, this graph is acyclic since under no circumstances a price is increased and therefore the Nash dynamics converge to a pure Nash equilibrium achieved by the SWM policy. Furthermore, each cloudlet derives the maximum possible spot price for which its utilisation criteria are satisfied, *i.e.,* $Y_s^d(\pi_{\text{init}}) = 0$ for all $d \in \mathscr{D}$. □

5.5.3 FogSpot Nash Equilibrium Derivation in Hierarchical Fog Topologies

Next, we focus on the RevM policy whose payoff function is:

$$\mathscr{R}_d^{\text{RevM}}(\pi) = \pi_d \sum_{s\in\mathscr{S}} Y_s^d(\pi)/\mu_s. \tag{5.10}$$

Unfortunately, there are no guarantees that, given a state π, the execution of the RevM policy will either decrease or increase its price.

Hence, we focus on the special case of fog topologies where cloudlets are deployed in a hierarchical way defined as:

Definition 3. *Two cloudlets are associated with a hierarchical relationship if the actions of the first do not affect the payoff of the second.*

In other words, a cloudlet d is hierarchically lower than a cloudlet d' if there is no class and LLA such that $u_{s,p}^d > u_{s,p}^{d'}$ while the relationship $u_{s,p}^d < u_{s,p}^{d'}$ is true for at least one class and LLA. In a sense, cloudlet d' is located closer to an end user meaning that by setting its spot price it determines the rate of LLA requests that serves; consequently, affecting the rate that requests left to get forwarded towards cloudlet d.

Two cloudlets can also be characterised by an uncorrelation relationship defined as:

Definition 4. *Two cloudlets are considered uncorrelated if their actions do not affect the payoff of each other.*

In detail, a cloudlet d is uncorrelated to a cloudlet d' if there is no class for any LLA that traverses both of them, *i.e.,* $\mathscr{P}_d \cap \mathscr{P}_{d'} = \{\emptyset\}$. Hence, the rates of provisioning each LLA to one of the two cloudlets, do not affect the provisioning rates at the other.

Definition 5. *A fog infrastructure is considered hierarchical if all the deployed cloudlets are associated either by a hierarchical or uncorrelated relationship.*

Theorem 2. *The FogSpot mechanism converges to a pure Nash equilibrium with the highest possible payoff, in the spot pricing game, when the RevM policy is applied in hierarchical fog topologies.*

Proof. Let the current state of spot prices be $\pi = (\pi_1, ..., \pi_D)$ and cloudlet d be the next that will apply the RevM policy. Without loss of generality, assume that cloudlet indexing imply the hierarchy in the topology, *i.e.,* only cloudlets 1 to $d-1$ have a hierarchical relationship with cloudlet d, meaning that their spot price affects the payoff of d. Then, if $d = 1$ the first execution of RevM policy by cloudlet will derive its optimal price π_1^* that is not affected by any other possible spot price in the fog, *i.e.,* $\mathscr{R}_1^{\text{RevM}}(\pi_1^*, ..., \pi_D) \geq \mathscr{R}_1^{\text{RevM}}(\pi)$ for all $\pi \in \Pi$. In general, if prices π_1 to π_{d-1} are optimal, for $d > 1$, then the newly derived price of d, π_d^*, is optimal in the sense that maximises its expected revenue since cloudlets 1 to $d-1$ will not change their price in the future,*i.e.,* $\mathscr{R}_{d'}^{\text{RevM}}(\pi_1^*, ..., \pi_d^*, ..., \pi_D) \geq \mathscr{R}_{d'}^{\text{RevM}}(\pi)$ for each $d' \in [1, 2, .., d]$ and all $\pi \in \Pi$. □

5.6 FogSpot Performance Evaluation

In this section, we demonstrate the performance of FogSpot in terms of average QoS gain, cloudlets' revenue, and cloudlets' idle time percentage. At first, we present a toy example of a single cloudlet and LLA with 10 classes, *i.e.,* 10 different QoS gain values, in order to shed light into the basic comparison between the SWM and RevM approaches. We analytically investigate the classes served by SWM and RevM under positive, negative, and no correlation between the QoS gain of a class and its demand. That is, we identify the LLA demand conditions that differentiate the RevM performance from the SWM one. In the second part of the evaluation, we focus on a small tree topology of 7 cloudlets placed over 3 latency levels. Then, we compare FogSpot against other approaches, namely Least-Frequently-Used (LFU), Self-Tuning, and Static provisioning, when applied over a small tree topology for an increasing number of VMs per cloudlet. In that way, we depict FogSpot's advantages against other approaches from the literature, which could be directly applicable in the setting of our problem, for different cloudlet serving capabilities. That is, the presented results of this section provide an adequate evaluation of FogSpot *i*) by capturing SWM and RevM behaviour under different demand conditions as well as cloudlet serving capabilities, and *ii*) by comparing against other candidate provisioning management approaches. We implement FogSpot and the baseline approaches by extending ICARUS caching simulator for application provisioning problems [160].

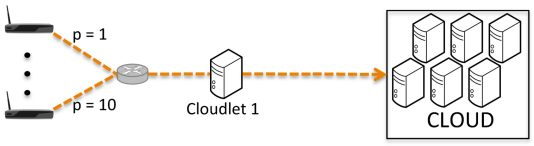


Figure 5.3: Toy Example Topology, Single cloudlet receiving requests from 10 classes.

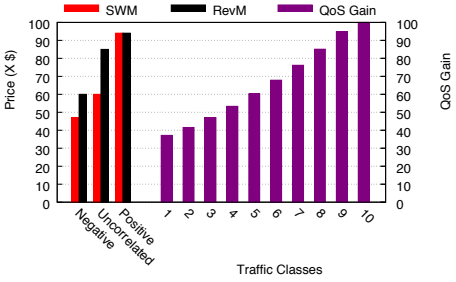


Figure 5.4: Toy Example, Spot Price for different demand-class correlations

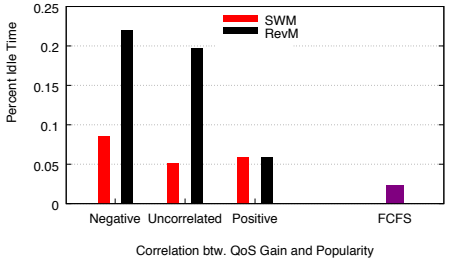


Figure 5.5: Toy Example, Idle Time of Resources for different demand-class correlation vs. FIFO.

5.6.1 FogSpot Toy Example

Here, we present a simple example that demonstrates the fundamental behaviour of FogSpot. In particular, we consider a setting of a single cloudlet, *i.e.,* Cloudlet 1, and a single LLA s whose requests fall into 10 classes, as depicted in Fig. 5.3. For the purpose of illustration, we associate class 10 to a QoS gain when provisioned at cloudlet 1 equal to 100, while the rest classes select a random QoS gain in the interval [30,100). That is, the QoS gains of LLA s classes are $\{37.15, 41.55, 47.03, 53.3, 60.25, 67.87, 76.15\ 85.11, 94.83, 100.0\}$. Lastly, we set the average engagement time of a user equal to 60 sec while the rate of arriving requests from all classes equal to 1, *i.e.,* $\sum_{p\in\{1,2,..,10\}} \lambda_{s,p} = 1$.

QoS Gain-Demand Correlation Experiments

We assign a capacity of 20 VMs to Cloudlet 1 and we demonstrate the behaviour of FogSpot in the following scenarios:

- Positive correlation of arriving requests and QoS gain, where the most popular class is Class 10, the second most popular is Class 9, *etc.*.
- Negative correlation of arriving requests and QoS gain, where the most popular class is Class 1, the second most popular is Class 2, *etc.*.
- Uncorrelated arriving requests and QoS gain, *i.e.,* a request has an equal probability of falling into any Class.

The requests correlation is expressed via a Zipf distribution of exponent 0.8 that categorises each request to one of the 10 classes by assigning its highest probability to Class 10 (Class 1) in the case

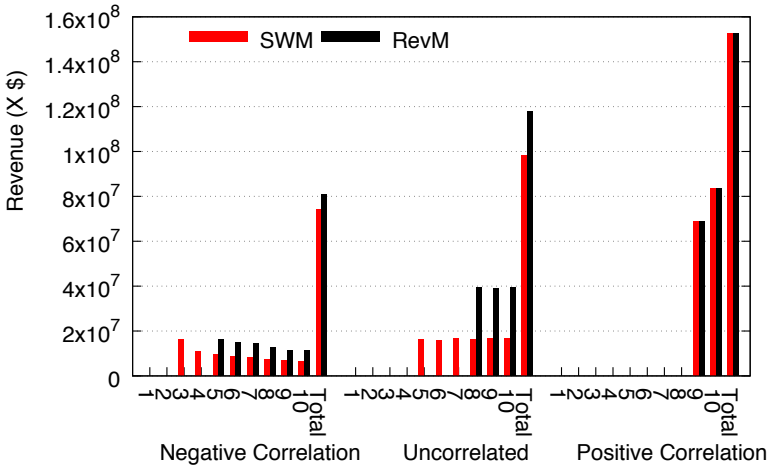


Figure 5.6: Toy Example, Per Class Revenue for different demand-class QoS Gain correlation.

of the positive (negative) correlation.

In Fig. 5.4, we plot the spot price achieved (in X $ per QoS gain unit[5]) by FogSpot for the 3 demand correlation scenarios. The RevM policy derives identical prices to the SWM policy for the case of positive correlation since the requests of classes 9 and 10 are sufficient for utilising the cloudlet resources, *i.e.,* the spot price is set equal to the QoS gain of class 9. That said, the RevM policy in the cases of uncorrelated and negatively correlated demand has the opportunity of trading resource utilisation for increasing the cloudlet revenue. In particular, in the advent of negative correlation (not correlation), the RevM policy sets a spot price that serves classes 5 to 10 (8 to 9) while SWM focuses on achieving a satisfying utilisation of the available VMs by accepting to also serve classes 3 and 4 (classes 5,6, and 7).

The tradeoff between resource utilisation and revenue increase is depicted in Figs. 5.5 and 5.6. In Fig. 5.5, the percentage of time a VM remains idle in RevM policy is 3 times (5 times) more than the corresponding SWM idle time of 8% (5%) for the case of negative correlation (no correlation). However, the idle percentage time in all cases and policies remains higher for the baseline approach that serves all classes, *i.e.,* the spot price equals 0, according to a FCFS policy. Lastly, the idle percentage time overhead of the RevM policy comes with the merits of a higher revenue as we see in Fig. 5.6.

Cloudlet Capacity Impact Experiments

Next, we focus on the case of uncorrelated arriving requests, *i.e.,* equal probability of request arrival for each class, and we illustrate FogSpot performance as we increase the number of VMs at cloudlet 1 of the toy example topology. Figures 5.7 and 5.8 show the impact of VMs number on the spot price and the effective rate of SWM and RevM policies. Clearly, the effective/admitted rate of SWM policy is strictly higher than that of RevM since in RevM each cloudlet trades its utilisation

[5]Without loss of generality, we assume a linear relation between prices and QoS gains.

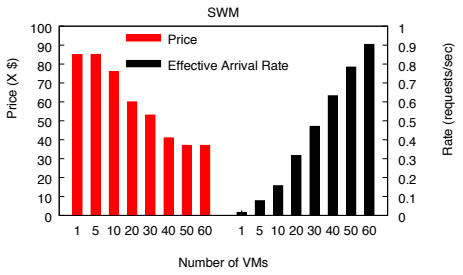


Figure 5.7: Toy Example, SWM policy: Spot Price vs. Effective Rate for an increasing number of VMs

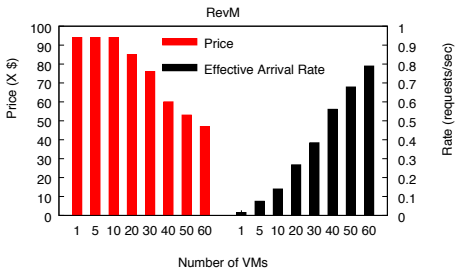


Figure 5.8: Toy Example, RevM policy: Spot Price vs. Effective Rate for an increasing number of VMs.

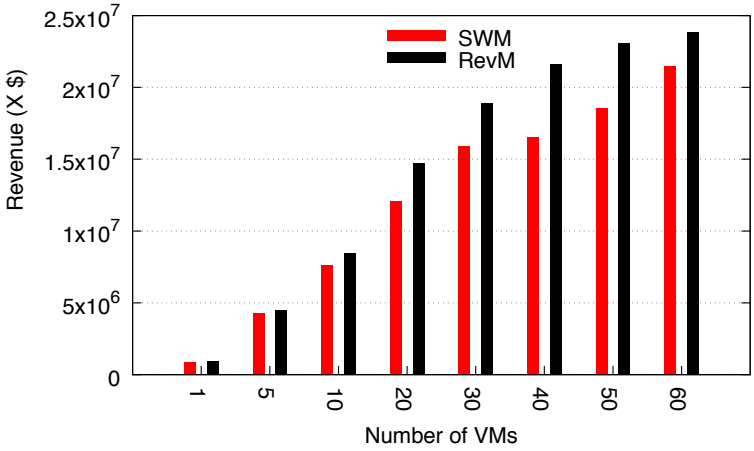


Figure 5.9: Toy Example, Revenue for different number of VMs.

of resources for a greater spot price that leads to a higher revenue as we see in Fig. 5.9.

5.6.2 Small Tree Topology

Consider the setting of Fig. 5.10 that depicts a small binary tree of three layers of seven cloudlets that create a round-trip-time to the cloud equal to 300 ms. We consider again the 10 LLA categories created according to the process described in section 4.4.1, but this time by setting the maximum latency equal to 300 ms. Similarly to the Edge-MAP evaluation, we compare FogSpot against a static, and a self-tuning approach. Furthermore, we also consider a Least-Frequently-Used (LFU) approach where the VMs of each cloudlet are allocated proportionally to each LLA's category popularity, that is monitored in time intervals of 10 minutes. The charging mechanism for these approaches also follow a spot price pay-as-you-go model in order to provide them the advantages of the FogSpot design, as apposed to charging them for the total period of utilising the resources. Specifically, the spot price is set to be equal to the QoS gain of the lowest LLA class served at a cloudlet. For simplicity, we assume that the average engagement of a user on average lasts for 1 minute. Lastly, at each leaf node, we generate one request per second according to a uniform probabilistic model, *i.e.,*

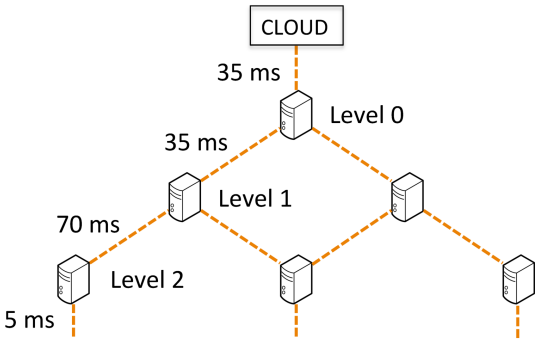


Figure 5.10: Small Tree Topology.

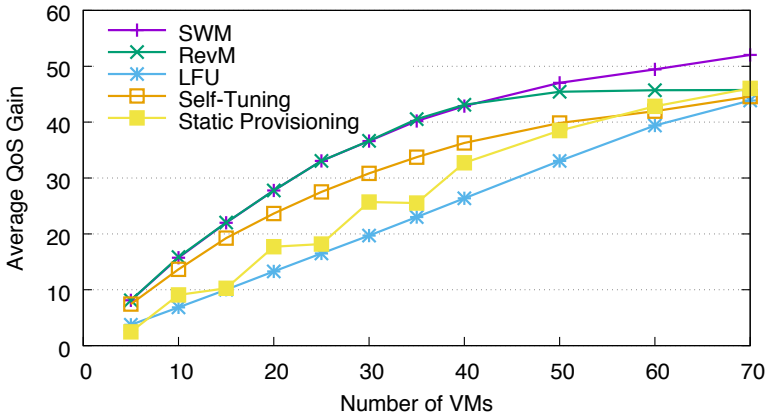


Figure 5.11: Small Tree Topology: QoS Comparison

each request select one of the 10 LLA categories with a probability equal to 0.1. We perform our comparison for an increasing number of VMs by executing 100 simulations of three hours duration for each setting.

Impact of the Number of VMs

In this experiment, we increase the number of VMs that are supported by each cloudlet of the small tree topology. In Figures 5.11- 5.13, we plot the main results of per-cloudlet VMs number increase in terms of QoS gain, Revenue, and Idle Time percentage. In particular, Fig. 5.11 depicts the QoS gain comparison of the described approaches. Clearly, the QoS gain increases function of VMs for all approaches, *i.e.,* the more VMs the better, with FogSpot SWM and RevM consistently be in better than the others by approximately more than 10%. Self-tuning approach is coming after, followed by Static provisioning which interestingly outperforms LFU. The reason is, that under the probabilistic requests' generation assumption, it is better to statically assign VMs according to their generation distribution instead of attempting to host the currently most popular LLAs, in terms of requests, as LFU does.

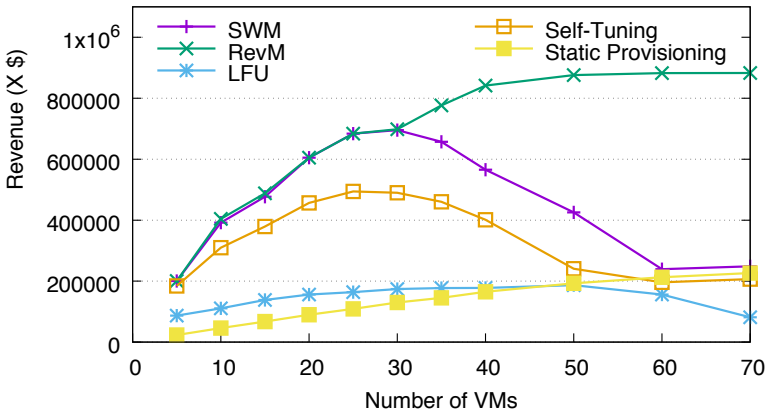


Figure 5.12: Small Tree Topology: Revenue Comparison

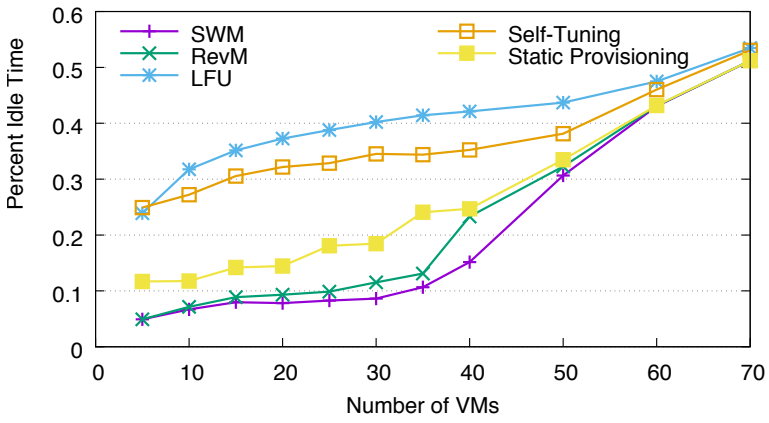


Figure 5.13: Small Tree Topology: Idle Time Percentage Comparison

Furthermore, SWM and RevM FogSpot policies have identical performance for up to the point of 30 VMs per cloudlet. After that, RevM starts trading Idle Time for increasing its revenue (Fig. 5.12) as opposed to SWM that aims the full utilisation of its resources, *i.e.,* the minimum possible idle time percentage (Fig. 5.13). In particular, RevM continues increasing its revenue, in terms of X$, while the revenue of SWM expectedly decreases. Specifically, for 70 VMs, the RevM has a revenue 4$\times$ higher than the other approaches. Regarding the other approaches, Static and LFU present the lowest revenues, since they do not consider the QoS gain for their LLA provisioning, while Self-tuning and LFU demonstrate the highest Idle Times, due to their reactive provisioning nature that does not take into account the requests' generation distribution.

A more detailed picture of how the idle time is propagated to the higher levels of the topology is depicted in Fig. 5.14. Each level has similar idle time for up to 35 VMs for both SWM and RevM. After that, in the RevM approach, the idle time starts increasing at Level 2, *i.e.,* leaf nodes, affecting

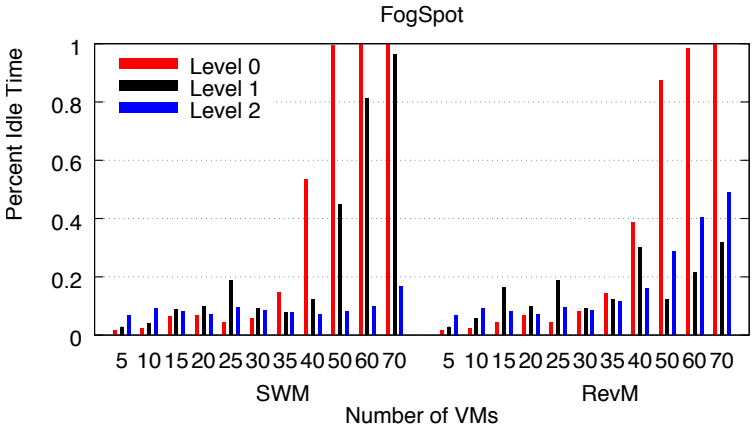


Figure 5.14: Small Tree, Number of VMs impact, Average Idle Time Percentage Per Layer, FogSpot SWM vs. RevM Comparison.

the higher levels of the tree which in comparison to the SWM approach present lower idle times, since more requests are rejected from Level 2 in RevM and get forwarded towards the cloud.

5.7 Conclusions

In this chapter we proposed FogSpot as an on-path, on-demand market based provisioning mechanism. FogSpot associates each cloudlet to a spot price that either targets to maximise the cloudlet's resource utilisation or revenue. After that, requests forwarded to the network interact with cloudlets' spot prices in a way that if their gain, in terms of QoS improvement, exceeds the spot price they provision an application's instance. FogSpot takes explicitly into account the provisioning of heavily stateful applications which once instantiated cannot be suspended in order to get migrated to another cloudlet location within a reasonable timeframe. Our game-theoretic analysis showed that FogSpot can derive the optimal spot prices in uncoordinated settings in hierarchical topologies. FogSpot advantages were demonstrated against a variety of proactive and reactive provisioning techniques over simple topologies.

Chapter 6

Enabling Complementary VNFs for LLAs

6.1 Introduction

Network functions inspect, filter, convert, and generally operate on applications' traffic for improving their security and performance [51]. Network functions typically come in the form of purpose-built intermediary hardware, *i.e.,* middleboxes, customised to perform specific tasks/functions for other purposes than packet forwarding. Service Function Chaining (SFC) refers to the process of steering a flow through a sequence of middleboxes that a flow has to traverse in the network before reaching its destination. That said, the process of steering flows through different network functions typically deviates the flows from their shortest path. In other words, SFC has a negative impact on the physical latency separating the client-server components, nullifying the QoS merits of LLAs' deployment over INCRs. Ideally, middleboxes should follow the dynamism of LLA instances over the INCRs; however, such an approach is infeasible due to the cost of middleboxes' purchasing [165] as well as the fact that middleboxes once setup cannot alter their structure (*e.g.,* topology) and functionality (*e.g.,* morph from one service to another).

Network Function Virtualisation (NFV) [56] has been proposed to increase the flexibility in network functions' usage, evolving middlebox architectures to virtual that can be deployed on top of commercial-off-the-shelf (COTS) hardware, *i.e.,* NFV nodes. NFV promises to improve the performance as well as the efficiency of network functions since both the structure and the (service) functionality of NFV nodes can be adjusted dynamically in response to network functions' demand. So far, research efforts have been focused on exploiting NFV in the context of data centres, for reducing capital and operational expenditures [198] of data centre providers, with some works even suggesting the offloading and virtualisation of ISPs' network functions to the cloud [172]. Nevertheless, in order to take advantage of NFV in complementing LLAs smooth operation, we have to consider INCRs capable of supporting virtual network functions, that we refer here as NFV nodes. In this chapter, we argue for the necessity of an algorithmic framework that incentivises third-party NFV nodes to jointly manage their Network Functions Instances (NFIs), in terms of instantiation/consolidation of virtual network functions, and take flow steering decisions in arbitrary network topologies, defined

by the location of NFV nodes.

Specifically, we propose a semi-DistRibutEd resource management framework for NFV based service function CHaining (**DRENCH**). DRENCH operates in the context of a regulated market that periodically associates each NFI to a price that is indicative of its workload. Then, based on the market assigned prices, the NFV nodes compete against each other by instantiating (consolidating) lucrative (unprofitable) NFIs in an effort of maximising their income. In addition to NFIs' instantiation/consolidation, each NFV node is also responsible for taking flow steering and redirection decisions when a flow requires additional network functions. We realise DRENCH in the context of SDN-NFV [128] architectures by defining a market environment of NFV nodes where the SDN controller acts as the market orchestrator/regulator among the participating NFV nodes. In DRENCH, the market orchestrator is setting the control parameters of *i*) *minimum/maximum NFIs' price* and *ii*) *off-path penalty factor*. The minimum NFI price defines a lower bound for consolidating NFIs whose prices are below the minimum one. Similarly, a maximum NFI price defines a threshold for estimating the additional number of NFI required to keep the price of an instance below this upper bound. Since NFIs' prices are representative of their workload, the minimum (maximum) NFIs' price indicates the thresholds below (above) which an NFI is considered being *under-utilised* (*over-utilised*), thereby controlling the number of active NFIs. On the other hand, the off-path penalty factor controls the path-stretch of a flow in the context of SFC, thereby penalising the choice of NFIs that force the flow to deviate from its shortest path towards the destination. Considering Flow Completion Time (FCT) as an index of flow performance, DRENCH minimum/maximum NFI price (off-path penalty factor) defines the tradeoff between under-utilised/over-utilised instances (flow path stretch) and FCT.

The main technical contributions of this Chapter involve:

- *A computationally feasible NFV management approach:* in DRENCH, resource management decisions are taken locally by NFV nodes while the market orchestrator solves lightweight problems, addressing a complex problem in a computationally feasible way with respect to *i) path-stretch, ii) number of active NFIs per service, iii) load on each NFI and iv) flow completion time*.

- *A decoupled NFV resource management framework for incentivising third-party NFV nodes:* in DRENCH, NFV nodes do not have to be owned by the same entity, as in other typical management approaches, contributing to the incremental adaptation of NFV in arbitrary network topologies for supporting LLAs.

- *Large scale evaluations:* We compared DRENCH in a simulation environment consisting of a Rocketfuel topology (87 switches) to a custom centralised approach: SIMPLE [152] on top of a E2 SDN framework [141].

6.2 Design Overview

6.2.1 Desired Properties

DRENCH is an in-network, congestion-aware, load balancing algorithmic framework that handles SFCs and dynamic NFIs in arbitrary network topologies. In designing DRENCH, we focus on providing the following key properties:

P1 **Efficiency:** As an NFI placement mechanism, DRENCH should neither under-utilise nor over-utilise the available computing resources of NFV nodes.

P2 **Cost awareness:** DRENCH should instantiate the minimum NFIs that meet the requirements of flows' SFCs at any point in time, and balance the utilisation among different NFIs.

P3 **Fine-grained flow handling:** DRENCH must meet each flows' SFC requirements, in terms of end-to-end latency and minimum throughput.

P4 **Responsiveness:** DRENCH should react to SFC traffic demand fluctuations, especially when traffic is volatile and bursty [109, 102] for arbitrary network topologies in order to support the dynamism of LLA deployment.

P5 **Incremental deployability:** DRENCH should require the minimum possible modifications in terms of protocols and network infrastructure. It should also be applicable to any of the existing SDN architectures [141, 88, 196] with minimal changes. Furthermore, it should be possible to directly apply DRENCH to a subset of available switches and/or incoming traffic when necessary.

6.2.2 DRENCH Solution Overview

The presented framework is designed to leverage the benefits of centralised as well as distributed networking paradigms. We use a **centralised approach**, *i.e.,* an SDN controller, to perform tasks with less computational load, but those that need to be carried out in a coordinated fashion across multiple nodes. These tasks include the: *i)* gather, compute and disseminate NFI load information periodically to all the decision making entities, *i.e.,* NFV nodes, and *ii)* set up paths towards instances and egress nodes in case they do not already exist. In addition, the SDN controller is used to decide which services are applicable to a flow (based on policies and/or flow characteristics). That is, the existence of a centralised controller reduces the complexity of the controller and also overcomes the issues faced by a purely distributed approach, where the decision making entities might not have up to date information, thereby degrading their performance. On the other hand, a **distributed approach** is used for decision making at individual NFV nodes. Based on the information provided by the controller, each node independently decides to *i) steer flows* towards the next required service; *ii) redirect flows* to the least loaded instance; and *iii) instantiate/terminate* NFIs in order to adapt to traffic demand. The high-level operation of the proposed mechanism is shown in Fig. 6.1.

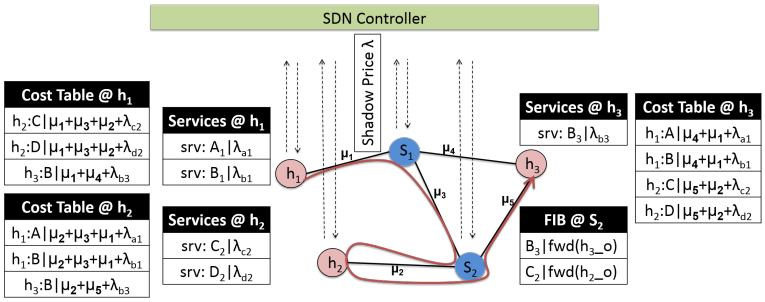


Figure 6.1: DRENCH High-Level Operation

6.3 DRENCH Components

DRENCH consists of the following components:

- **Market Orchestrator:** It associates every NFI and link resource to a *shadow price* (*i.e.,* cost) produced by utilising global information available at the controller. The orchestrator regulates the market by allowing the existence of instances above a certain minimum price.
- **Flow Steering and Redirection:** This component steers each flow through a valid sequence of NFIs (according to its SFC) determined by the SDN controller. Flow steering and redirection takes into account the flow steering latency as well as the NFIs' and links' utilisation.
- **NFI Instantiation/Consolidation:** This component instantiates and consolidates NFIs in a distributed way through market competition between NFV nodes.

Below, we describe each of these components in detail.

6.3.1 Market Orchestrator

DRENCH, as any market-based approach, requires the association of each network resource (commodity), in terms of NFIs and link bandwidth, to an offered price, which is imposed on a given set of incoming flows (demand) that utilise this resource. In particular, when the quantity of demanded resources equals the quantity supplied for a set of prices, we refer to them as *market-clearing prices*. DRENCH market-clearing prices should *A1*) be representative of each NFI's workload, *A2*) be derived in the shortest possible time, and *A3*) not require additional in-network signalling given the existence of an SDN controller [111, 26]. Every price derivation violating requirements (*A2*) and (*A3*) would be in stark contrast with DRENCH desired properties *wrt* responsiveness (***P4***) and incremental deployability (***P5***), respectively.

DRENCH deploys a *Market Orchestrator/Regulator* component, which by simply exploiting the already available at the SDN controller information about the path of each flow, efficiently derives the market-clearing prices while complying to requirements (*A1*)-(*A3*). Inspired by [113], where the authors formulate a Network Utility Maximisation problem (NUM) based on market principles to allocate bandwidth resources to a set of flows, we extend their model to include NFI computational resources. We achieve this by solving the Extended Network Utility Maximisation problem (ENUM) at the Market Orchestrator as we describe next.

$\mathscr{G}$	Network topology
$\mathscr{V}$	Set of switches
$\mathscr{E}$	Set of links
$\mathscr{H}$	Set of NFV Nodes
$\mathscr{S}$	Set of services
$\mathscr{H}_s$	Set of NFV nodes executing service s
$\mathscr{F}$	Set of flows
x_f, x_f^*	Rate and optimal rate of flow f
$U_f(x_f)$	Utility function of flow f
b_e	Bandwidth capacity of link e
$a_{e,f}$	Coefficient = 1, if flow f traverses link e
b_s^h	Computational resources of NFI s at h
d_s	Computational power Required by a NFI of s for processing a single bit of traffic
$d_{s,f}^h$	Coefficient = d_s if f is processed by NFI s at h
w_f	Weight of flow f
λ_s^h	Service cost of NFI s at NFV node h
$\underline{\lambda}, \bar{\lambda}$	Minimum and Maximum shadow prices, defining the efficiency of an instance
p_{v_i,v_j}	Shortest path from switch v_i to switch v_j
μ_{v_i,v_j}	Communication cost from switch v_i to switch v_j
$\mathscr{C}_{v_i,h}(s)$	Communication and service cost from switch v_i to a NFI executing service s at NFV node h
$\lvert p_{v_i,v_j} \rvert$	Number of Hops from switch v_i to switch v_j
$\Delta p_{v_i,h}^f$	Shortest path deviation overhead
ρ	Off Path penalty factor
$\mathscr{C}_{v_i,h}^f(s)$	Estimated $\mathscr{C}_{v_i,h}(s)$ cost of flow f including ρ
θ_{rid}	Redirection threshold
P_h	Profit of NFV h in terms of shadow prices
$\widetilde{\lambda}_{on}, \widetilde{\lambda}_{off}, \widetilde{\lambda}$	On-/Off- path and expected competitive price

Table 6.1: DRENCH Notation

We denote the network topology by $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, of $\mathscr{V}$ switches and $\mathscr{E}$ links, where a set of NFV nodes, $\mathscr{H}$, is placed at a subset of switches $\mathscr{H} \subseteq \mathscr{V}$. Then, given a set of NFIs executing a set of $\mathscr{S}$ services and a set of $\mathscr{F}$ flows, we associate each link, $\forall e \in \mathscr{E}$, with a bandwidth capacity, b_e, and each NFI s at NFV node h with b_s^h computational resources, in order to form the ENUM problem that maximises the total utility of the system. Similar to NUM, we associate each flow rate, $x_f \geq 0$, with a utility that is a weighted logarithmic function, $U_f(x_f) = w_f \log(x_f)$, of weight w_f, capturing a decreasing marginal gain as the flow rate increases (*i.e.,* rate changes at low rate flows have a greater impact on their utility). In turn, we maximise the total system utility, $\sum_{f \in \mathscr{F}} U_f(x_f)$, subject to link capacity constraints, $\sum_{f \in \mathscr{F}} a_{e,f} x_f \leq b_e$, $\forall e \in \mathscr{E}$, and computational resource constraints, $\sum_{f \in \mathscr{F}} d_{s,f}^h x_f \leq b_s^h$, $\forall s \in \mathscr{S}, \forall h \in \mathscr{H}$; where $a_{e,f}$ is a coefficient equal to 1 if flow f traverses link e and 0 otherwise, while $d_{s,f}^h$ equals the computational power required by service s for processing a single bit of traffic, d_s, if f is executed at NFI s of NFV node h, and 0 otherwise. Parameters $a_{e,f}$ and $d_{s,f}^h$ describe the path of each flow and therefore they are known to the SDN controller which provides them to the Market Orchestrator.

Since the objective function is differentiable and strictly concave, while the feasible region of the constraints is compact, the optimal rates $x_f^* \; \forall f \in \mathscr{F}$ exist, are unique, and can be found efficiently by Lagrangian methods. Based on [113], it can be shown that the dual problem of the ENUM is:

$$\begin{aligned} \text{maximise} \sum_{f \in \mathscr{F}} w_f \log \Big(\sum_{e \in \mathscr{E}} \mu_e a_{e,f} + \sum_{h \in \mathscr{H}} \sum_{s \in \mathscr{S}} \lambda_s^h d_{s,f}^h \Big) \\ - \sum_{e \in \mathscr{E}} \mu_e b_e - \sum_{h \in \mathscr{H}} \sum_{s \in \mathscr{S}} \lambda_s^h b_s^h \end{aligned} \tag{6.1}$$

subject to

$$\begin{aligned} \mu_e \geq 0, \; \forall e \in \mathscr{E}, \\ \lambda_s^h \geq 0, \; \forall s \in \mathscr{S}, \forall h \in \mathscr{H}, \end{aligned}$$

where μ_e and λ_s^h are the Lagrange multipliers of link e and service instance s at NFV node h, respectively. The Lagrange multipliers are also known as *shadow prices*, due to their association to the optimal rates of each flow:

$$x_f^* = \frac{w_f}{\sum_{e \in \mathscr{E}} \mu_e a_{e,f} + \sum_{h \in \mathscr{H}} \sum_{s \in \mathscr{S}} \lambda_s^h d_{s,f}^h} \tag{6.2}$$

where weight w_f is perceived as the budget that flow f is willing to pay for its rate, while the denominator is the cost imposed to the flow in order to use the resources along its path. In that sense, *each Lagrange multiplier can be considered as the price of a particular resource*, leading us to the following definition about communication and service cost.

Definition 6. *The communication cost between two switches, $v_i, v_j \in \mathscr{V}$, is the sum of on-path link shadow prices $\mu_{v_i,v_j} = \sum_{e \in p_{v_i,v_j}} \mu_e$, where p_{v_i,v_j} is the shortest path between switches v_i and v_j; while the service cost of an instance s at NFV node h is the shadow price λ_s^h.*

Note that the service and communication costs are kept in the forwarding tables of the NFIs, *i.e.,* the decision making nodes, and are updated periodically by the SDN controller after being estimated by the Market Orchestrator (see Fig. 6.1).

Shadow prices are indicative of the workload at a particular resource, complying to (*A1*). In fact, from (6.2), we derive that the value of a shadow price, λ, defines the maximum possible rate that flows using that resource can achieve, w_f/λ.[1] Based on the maximum achievable flow rate we can define the efficiency of a NFI as a range of shadow prices.

Definition 7. *The Market Orchestrator determines the load of a NFI by a shadow price range* $[\underline{\lambda}, \bar{\lambda}]$*, where if a service cost,* λ_s^h*, is less (more) than* $\underline{\lambda}$ $(\bar{\lambda})$*, the NFI is considered under-utilised (over-utilised), respectively.*

Given the shadow price range $[\underline{\lambda}, \bar{\lambda}]$, the Market Orchestrator tries to maintain the minimum required number of instances per service type (***P2***) by: *i)* terminating instances that are underutilised and *ii)* allowing for more NFIs whose existing instances are over-utilised (see Section 6.3.3).

6.3.2 Flow Steering and Redirection

Flow Steering

Given the placement of NFIs and their respective shadow-prices, as determined by the Market Orchestrator, DRENCH's flow steering component is responsible for steering each new incoming flow towards the chain of required NFs. The flow steering component tries to route each flow through the chain that imposes the lowest possible cost to the flow. However, determining the optimal end-to-end path of a flow through the SFC is a NP-complete problem [64]. DRENCH works on a hop-by-hop heuristic basis, picking each time the best next-hop NFI choice, in an effort to achieve instantaneous and adaptive steering decisions (***P4***).

We illustrate DRENCH's flow steering component through the following example. Assume that flow f arrives at the network requiring the execution of service s (or service chain $s_1/s_2/.../s_m$) before being delivered to destination v_f. Let v_i be the switch that has to make a steering decision about f and $\mathscr{H}_s \subseteq \mathscr{H}$ be the set of NFIs of service s. Then, the combined communication and service s execution cost at $h \in \mathscr{H}_s$ is $\mathscr{C}_{v_i,h}(s) = \mu_{v_i,h} + \lambda_s^h$. The flow steering component initially estimates the shortest path deviation overhead applied by steering flow f to instance h in terms of hops, *i.e.,* $\Delta p^f_{v_i,h} = |p_{v_i,h}| + |p_{h,v_f}| - |p_{v_i,v_f}|$, before weighting the deviation by an *off-path penalty factor* ρ. Therefore, the estimated cost applied to the flow for executing service s at NFI of h is $\mathscr{C}^f_{v_i,h}(s) = \mathscr{C}_{v_i,h}(s) + \rho \Delta p^f_{v_i,h}$. Then, v_i selects the next service instance s of f that minimises $\mathscr{C}^f_{v_i,h}(s)$:

$$h^* = \underset{h \in \mathscr{H}_s}{\operatorname{argmin}} \mathscr{C}^f_{v_i,h}(s) \tag{6.3}$$

[1] It follows that the shadow prices are positive when a resource is totally utilised and 0 otherwise. To introduce a minimum workload to the resources that are not saturated, we add a set of *dummy flows* into $\mathscr{F}$ when solving (6.1).

In Eq. (6.3) the off-path penalty factor, ρ, dis-incentivises node v_i from sending flow f away from its shortest path towards v_f. Eq. (6.3) applies on a hop-by-hop basis, that is, it is calculated at each NFV node responsible for forwarding flow f towards the next instance in its chain.

Lastly, upon making a steering decision, switch v_i informs the SDN controller that flow f is forwarded towards h^* to execute service s. At the same time, the SDN controller is setting up paths towards NFIs and/or egress nodes as necessary.

Flow Redirection of Stateless and Stateful flows

The cost of a service instance might change dramatically throughout the duration of a flow, rendering previous flow steering decisions outdated. Therefore, *redirection of existing flows* is necessary in order to keep the expenditure of existing flows at low levels (***P3***-***P4***) and avoid routing through overutilised instances (***P1***). We realise flow redirection as follows: if the cost difference between two instances of s at h and h', as seen by switch v_i, is bigger than a redirection threshold, θ_{rid}, $\mathscr{C}_{v_i,h}(s) - \mathscr{C}_{v_i,h'}(s) > \theta_{rid}$, switch v_i repeats the flow steering process for a portion of flows that v_i currently forwards to h. The redirection threshold is set to $\theta_{rid} = \bar{\lambda} - \underline{\lambda}$.

Rerouting of stateful flows to dynamically instantiated services for improving load balancing is usually complex and costly. For instance, solutions such as Split/Merge [154], pause ongoing flows in order to transfer internal NF and forwarding states. In DRENCH, we leverage the approach in Split/Merge [154], to pause ongoing flows and transfer the internal state of the involved network functions. To identify service instances, we assume the underlying deployment of a Information Centric Networking (ICN) construct which is proven to be beneficial in terms of providing flexible routing, and reducing the routing states at the switches [31].

6.3.3 Instantiation

In DRENCH, NFV nodes autonomously provide NFs in an effort to maximise their profit, in terms of *shadow prices* (*i.e.,* the cost to execute a NF). In particular, let S_h be the set of NFIs at some NFV node h, then the profit of h in terms of *shadow prices*, λ_s^h, can be estimated, as:

$$P_h = \sum_{s \in \mathscr{S}_h} \lambda_s^h \tag{6.4}$$

DRENCH NFI instantiation/consolidation scheme defines how service demand and NFI *shadow prices* affect the individual NFV node decisions to manage the number of service instances. Through competitiveness, NFV nodes achieve responsiveness to NF demand changes, while the market orchestrator ensures market efficiency, as we explain next.

NFV Node Competitiveness

Let the *shadow price* of an NFI s at NFV node h' be λ'. We are interested in estimating the competitive price of a potential NFI s at an NFV node h, $h \neq h'$, with respect to λ'.

Definition 8. *The shadow price of NFI s at h is competitive to the price of NFI s at h', λ', when the flow steering component has a preference, or is indifferent, of steering new flows at s of h.*

Then, let μ be the communication cost, between NFV node h' and h that are Δp hops away, and f be a new flow that is about to get steered at NFI s of NFV node h'. Then according to DRENCH's flow steering component, the minimum competitive price of s at NFV node h for flow f, would be equal to $\widetilde{\lambda}_{off} = [\lambda' - \mu - \rho\Delta p]^+$, where ρ is the off-path penalty factor.[2] The off-path penalty factor is taken into account as in the worst case that flow f will have to deviate by Δp hops to reach node h from h'. This acts as a disincentive for a node to forward traffic to nodes that are far off from the flow's shortest path. On the other hand, in the best case, that flow f is forwarded to NFV node h' via NFV node h, meaning that h is already *on the path* of flow f and additional hops are not required;[3] the minimum competitive price at s for flow f is $\widetilde{\lambda}_{on} = \lambda' + \mu$.

The expected competitive price of NFI s at h with respect to the corresponding NFI price at h', λ', will be a value between $(\widetilde{\lambda}_{off}, \widetilde{\lambda}_{on})$. Let y be the total amount of traffic with competitive price $\widetilde{\lambda}_{on}$. Then y can be considered as the local information of the competitive NFI demand at NFV h that accounts for the utilization percentage $d_s y / b_s^{h'}$. Here, d_s is the computational power required by the service of NFI s for processing a single bit of traffic and $b_s^{h'}$ is the fixed computational resources that are allocated to NFI s at h'. Then, the expected competitive price is estimated as:

$$\widetilde{\lambda} = (d_s y / b_s^{h'})\widetilde{\lambda}_{on} + (1 - d_s y / b_s^{h'})\widetilde{\lambda}_{off} \tag{6.5}$$

NFI Instantiation

As long as a NFI shadow price, λ, executing a service at NFV node h, is lower than the maximum target price, $\lambda < \bar{\lambda}$, this service is not considered over-utilised and an instantiation of an additional NFI of the same service at h is prohibited (see also *Definition 2*). On the other hand, if $\lambda > \bar{\lambda}$, the Market Orchestrator limits the number of NFIs that can be created by competing the NFI with service cost λ to $\lfloor \lambda / \bar{\lambda} \rfloor$. Therefore, given the set of allowed services for instantiation at each NFV node h, h estimates the expected competitive prices of every NFI. Then, moving from the highest to the lowest competitive price, the NFV node instantiates the service associated with the price $\widetilde{\lambda}$ as long as *i*) it is expected that the instance will not be under-utilised, $\widetilde{\lambda} > \underline{\lambda}$, and *ii*) the Market Orchestrator maximum number of instances allows it, respecting properties (***P1***), (***P2***), and (***P4***).

NFI Consolidation

If the price of an instance is below the minimum target shadow price, $\underline{\lambda}$, the NFV node consolidates this instance ***(P2)***. When there is a service availability requirement, the market orchestrator can hinder the consolidation of the last instance of that service.

6.4 DRENCH Evaluation

In this section, we evaluate DRENCH's performance in terms of *i*) Path Deviations, in number of hops, and *ii*) Flow Completion Time (FCT), in seconds required to complete the transmission of a

[2] $[\cdot]^+$ denotes the projection onto nonnegative orthant.

[3] In practice, it is not the NFV node that is aware of the forwarded traffic but the switch that the NFV node is attached to.

Arrival rate	100 flows per second
Off-path penalty, ρ	0.3
$\underline{\lambda}$	0.1
$\bar{\lambda}$	0.3
Flow size	10 Mbytes
Stateful NFIs, No Redirections	True
NFIs processing capability	1 Gbps
Initial number of NFIs per service	1
Maximum number of NFIs in the Network	57
Service Chain Length	2
Total Number of Services	8
Flow Generation Interval	5 minutes
Experiment Duration	10 minutes

Table 6.2: DRENCH Default Evaluation Setting

flow. In detail, we investigate the impact of DRENCH's control parameters on the trade-off between path deviation and FCT over an ISP topology, before proceeding to the comparison of DRENCH against a customised semi-distributed NFV management approach. We implement DRENCH on a python-based discrete event simulator using *SimPy*[4].

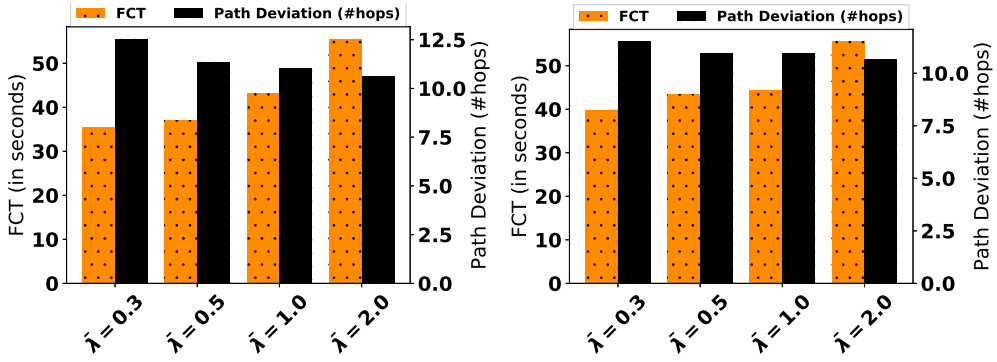
6.4.1 DRENCH Experimental Setup

We deploy DRENCH over the Rocketfuel AS-1755 (Ebone in Europe) topology[5]. In this ISP topology, we define 27 hosts that send/receive flows and 57 nodes that are capable of supporting a NFI that can process up to 1.5 Million packets per second, equivalent to 1 Gbps of traffic[6]. Furthermore, in order to capture DRENCH's adaptiveness in the case of burst traffic conditions, we model the traffic as *elephant* flows [108] of 10 megabytes generated at the rate of 100 flows per second. We evaluate DRENCH's performance over a period of 10 minutes while we generate flows only for the first 5 minutes, *i.e.,* in total we generate 30,000 flows. Lastly, based on [176] we setup a service function chain of 2 distinct Network Functions while in total we consider 8 NFs selected in distinct pairs with equal probability, *i.e.,* $\mathscr{S} = \{\text{A,B,C,D,E,F,G,H}\}$ where a service chain has equal chances to be either AB, CD, EF, or GH. Initially, each network function has a single instance that we place at the topology starting from the nodes with the highest betweenness centrality, *i.e.,* the higher the number of shortest paths passing from a node, the higher its centrality is in order to impose a low

[4]http://simpy.readthedocs.io/en/latest/

[5]`http://www.cs.washington.edu/research/projects/networking/www/rocketfuel/interactive/1755eur.html`

[6]Assuming an Intel Xeon Processor E5-2600 Family Core 1C@1.3 GHz [70].



(a) $\underline{\lambda} = 0.1$ (b) $\underline{\lambda} = 0.3$

Figure 6.2: Maximum shadow price tradeoffs for $\bar{\lambda} = 0.3, 0.5, 1.0, 2.0$ in two scenarios

path deviation in case that there is no need for additional instances. The default evaluation setting is presented in Table 6.2.

6.4.2 DRENCH's Parametrisation

Shadow Prices Range

First of all, we study the impact of minimum and maximum shadow prices on DRENCH's performance. Specifically, since we are interested in evaluating only the influence of price parameters we set the off-path penalty factor equal to 0. In Fig. 6.2 we plot the tradeoff between the FCT and path deviation as the maximum shadow price increases for two different minimum shadow prices, namely $\underline{\lambda} = 0.1$ (Fig. 6.2a) and $\underline{\lambda} = 0.3$ (Fig. 6.2b). An increase of $\bar{\lambda}$ is equivalent to *i*) higher permitted utilisation of existing NFIs, and *ii*) conservative instantiation of new NFs. In other words, the minimum target processing rate for each flow decreases while the number of NFIs that are allowed to be instantiated, as a response to an over-utilised NFI, declines. That is, a higher maximum shadow price, $\bar{\lambda}$ leads to increased FCTs, due to the lower allowed processing rate per flow, and reduced average path deviations, since the newly instantiated NFs are starting by occupying nodes of higher centrality and their allowed instances limitation eventually imposes a lower path deviation, as depicted in Fig. 6.2 for both prices of $\underline{\lambda}$.

On the other hand, a higher minimum shadow price, $\underline{\lambda}$, decreases the maximum permitted processing rate for each flow while keeping the utilisation of NFIs to higher levels. Overall, by increasing both the minimum and maximum allowed shadow prices, the average path deviation decreases at the expense of FCT that rises, as we see in Figs 6.2a-6.2b. The exact setting of $\underline{\lambda}$ is up to the network operator. If demand is low, then more instances should be allowed to reduce the average FCTs. On the other hand, during high demand periods, the operator might have to compromise on individual FCT, in order to fully utilise the existing NFIs and eventually serve more flows overall. We set $\underline{\lambda} = 0.1$ and $\bar{\lambda} = 0.3$ for defining the interval of ideal NFI utilisation as the

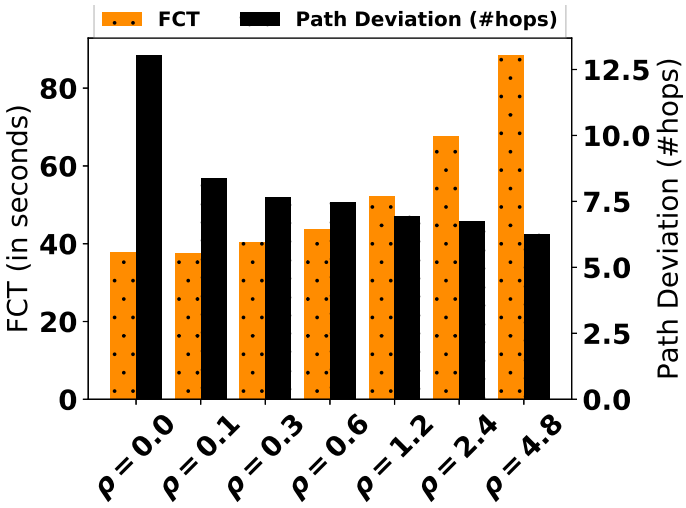


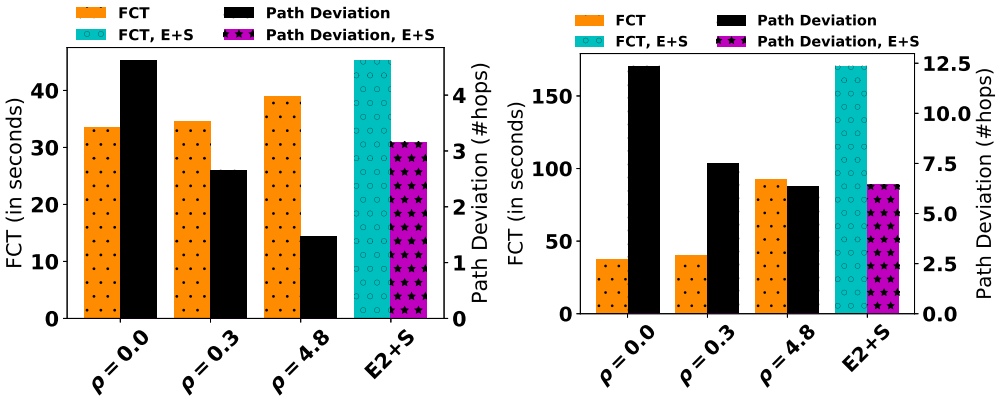
Figure 6.3: Off-path penalty factor, ρ, impact

combination that achieves the lowest FCT.

Off-path Penalty

In the context of service chaining, flows deviate from their shortest path in order to be served by NFIs. In DRENCH, the off-path penalty factor, ρ, controls the tradeoff between the shortest path deviation and FCT, by *trading* the overhead of path deviation for less utilised NFIs, as Figs. 6.3 indicates. In more detail, the FCT increases function of off-path penalty, ρ, since flows prefer to get served by a more congested NFI, *i.e.,* with a higher service price, than deviating from their shortest path. Specifically, by setting the off-path penalty factor equal to 0 we always steer the incoming flows through the least utilised NFIs without considering the aspect of path deviation, resulting in 12.5 hops of path deviation in our setting. Interestingly although expectedly, by slightly increasing the off-path penalty factor to 0.1 we achieve a much lower path deviation, *i.e.,* 7.5 hops, without degrading the FCT, compared to the one achieved for $\rho = 0$; this result is indicative of the off-path factor importance in flow steering decisions.

The exact setting of the ρ factor is up to the network operator. During low-demand periods where links are generally less utilised, or for bulk traffic where latency is not important, *e.g.,* software updates, operators might choose a lower value to improve the involved FCT since the extra path deviation is not degrading the network performance for no-latency-sensitive flows. On the other hand, during periods of high demand and/or latency sensitive traffic (related to LLAs), path deviation should be kept to lower levels even if this increases the individual FCTs. As a default value, we select $\rho = 0.3$ since it achieves a fair balance of FCT and path deviation.



(a) Service Chain Length 1

(b) Service Chain Length 2

Figure 6.4: DRENCH comparison for different off-path penalty factors against E2+S.

6.4.3 DRENCH's Comparison

Next, we compare DRENCH against the state-of-the-art in distributed NFV management and service function chaining. Specifically, since DRENCH is the first framework that tackles both problems, we have to combine 2 separate frameworks namely *i*) E2 [141], that is specialised in instantiating/consolidating NFIs, and *ii*) SIMPLE [152], which performs flow steering based on the workload of each service function chain. Specifically, E2 has been designed for deployment into data centres and its NFV management strategy involves the instantiation of a new network function, in response to an over-utilisation signal, to the *closest* possible available location, *i.e.,* NFV node with spare capacity. On the other hand, SIMPLE aims to minimise the maximum VNF load across the network by solving a linear program which determines the fraction of traffic forwarded to each preselected SFC. We refer to the combined approach where the NFV management takes places according to E2, while the flow steering decisions are managed by SIMPLE as E2+S. E2+S relies on $\underline{\lambda}$ and $\bar{\lambda}$ for the signalling of over- and under-utilised NFIs that again are becoming available by the market orchestrator, although E2+S is based on the assumption that the NFV nodes belong to a single entity, *i.e.,* they do not compete with each other.

In Fig. 6.4, we compare DRENCH for different off-path penalty factors, *i.e.,* $\rho \in \{0.0, 0.3, 4.8\}$, against E2+S for service chain lengths of 1 (Fig. 6.4a) and 2 (Fig. 6.4b). For a service chain of a single service, E2+S is outperformed by DRENCH for $\rho = 0.3$ and $\rho = 4.8$ both in terms of FCT and path deviations. However, for $\rho = 0.0$ DRENCH is worse than E2+S by 30% in path deviation although it has a lower FCT by 10 seconds. On the other hand, for a service chain length of 2 services E2+S returns a fairly low path deviation that is 50% smaller than the corresponding one of DRENCH for $\rho = 0.0$, at the cost of FCT, that is more than 400% greater than DRENCH for $\rho = 0.0$. However, DRENCH outperforms E2+S for $\rho = 4.8$ where it achieves half of the FCT, *i.e.,* 80 seconds instead of 170, at the same path deviation overhead, *i.e.,* almost 6 extra number of hops.

6.5 Conclusion

Network Function Virtualisation evolves middlebox architectures to virtual enabling the deployment of network functions on top of commercial of-the-self hardware. That is, network functions can change their functionalities as well as structure according to traffic demand. In this chapter, we investigated the deployment of virtual network functions over INCRs in arbitrary topologies. This is a promising research direction that guarantees LLAs' smooth operation by assuring the support of VNFs at the edge and middle-tier locations of the network.

We proposed DRENCH, a semi-distributed resource management framework for NFV-based service function chaining, which operates in the context of a regulated market. In detail, a market orchestrator associates each active NFI to a price that reflects its workload levels. After that, NFV nodes compete with each other for hosting the most profitable NFIs, in a distributed way, while they are also responsible for taking flow steering decisions. Our evaluation indicates DRENCH's capability in capturing the tradeoff of FCT and path deviation.

Chapter 7

Conclusions and Future Research Directions

7.1 Summary

Current practices in deploying network applications' involve the provisioning of computing resources into distant data centres. That said, an increasing number of Low Latency Applications, defined as the network applications that require much lower response times than the ones supported by the data centres, render typical provisioning approaches unfit for purpose. At the same time, there is consensus about the deployment of computing resources in middle-tier locations and the edge of the network. In this thesis, we argued about the provisioning of Low Latency Applications over In-Network Computing Resources, deployed *closer* to the end users, that can improve the QoS of LLAs by supporting lower round trip times. Specifically, we investigated the deployment of Low Latency Applications over third-party In-Network Computing Resources. Along these lines, we developed a set of frameworks for enabling LLAs over INCRs under different connectivity scenarios as we summarise next.

First of all, in Chapter 3, we presented a quality assessment framework for streaming applications under disrupted connectivity. Our framework contributes to crowdsourcing-based applications' delivery, in the absence of connectivity, by quantifying the performance of different application parameters' configuration (*e.g.,* number of streaming channels, battery consumption preferences *etc.*) in different connectivity settings (*e.g.,* pattern of disconnection intervals). In the particular case, a streaming application provider can compare different configuration options in order to improve her performance. The framework's usage as a feasibility tool was demonstrated in the realistic setting of urban railway networks where, by analysing real commuter traces, we quantified the gain of downloading content collaboratively, in terms of undisrupted playback time, for dealing with connectivity disruptions caused by the trains' movement between stations.

After that, in Chapter 4, we introduced Edge-MAP, a market based mechanism for provisioning LLAs over INCRs for mobile users. Edge-MAP contributes in presenting the first auction-based LLA provisioning mechanism that relies on cellular-based markets with respect to end users' mobility. At a micro-level, Edge-MAP leverages Vickrey-English-Dutch auctions to map mobile users'

requests, at each cellular market, to offered VMs in a robust way. At a macro-level, Edge-MAP fosters competition among the providers of INCRs by enabling a feedback mechanism with respect to profit opportunities on different markets. Edge-MAP is unique in its design and, to the best of our knowledge, it is the first to exploit the gains of Vickrey-English-Dutch auctions in environments where demand and supply conditions evolve over time. We demonstrated Edge-MAP's design choices on realistic vehicular traces that we considered as mobile users.

In Chapter 5, we proposed FogSpot as another market mechanism for provisioning LLAs over INCRs but this time under fixed connectivity conditions. FogSpot is designed to take into account the provisioning of heavily stateful LLAs, that once instantiated for serving a user, they cannot be disrupted and migrate to another INCR without failing. In FogSpot, INCRs offer their available VMs via collocated markets that interact with forwarded users' application requests in real time. FogSpot associates each cloudlet with a spot price based on the applications requests' rates. FogSpot is the first proposed charging mechanism for on-path, on-demand, LLA provisioning over INCRs.

In Chapter 6, we designed a semi-distributed resource management framework for NFV based service function chaining, *i.e.,* DRENCH, that is designed to be applied on arbitrary network topologies, as opposed to data centres' fat-tree ones. DRENCH operates in the context of a regulated market that associates each NFI to a price that is representative of its workload. After that, the NFV nodes, *i.e.,* the INCRs that are capable of supporting virtual network functions, instantiate and consolidate NFIs through competition. DRENCH contributes uniquely in enabling NFV over arbitrary topologies that consist of third-party INCRs in an economic context. These facts render DRENCH ideal for managing Virtual Network Functions that complement the deployments of LLAs.

In summary, this thesis investigated the management of INCRs for supporting LLAs from a multitude of viewpoints. Our ultimate goal was to provide the mechanisms for provisioning LLAs in an economic context that compensates, and therefore incentives, the participation of INCRs. We consider the presented solutions as the most promising research direction for deploying LLAs since they are *i*) immediately applicable, *i.e.,* can be applied on existing resources, *ii*) incrementally deployable, *i.e.,* INCRs' infrastructure can be expanded over time and *iii*) sustainable, since INCRs can continue being used as an infrastructure for deploying LLAs only if the application provisioning mechanisms operate in an economic context that compensates them.

7.2 Future Work

The presented work of this thesis is setting the foundations of the future Internet as a platform for accessing applications in a seamless, ubiquitous, and network-performance-aware way. This ultimate vision involves both users and machines equipped with multiple devices that remain seamlessly connected to a plethora of diverse applications, with diverse requirements, in smart environments. However, such a vision requires the ubiquitous deployment of an alternative computing infrastructure that remains accessible under different connectivity scenarios. Unfortunately, the cost of such

an universal infrastructure is prohibitive, while the absence of such an infrastructure inhibits the development of applications that otherwise would contribute to the proliferation of automative and IoT domains; as a result, the Internet's role as a platform that brings together users and applications is currently undermined.

This thesis investigates the promising direction of forming the required alternative computing infrastructure by exploiting existing in-network computing resources. In particular, we argue about the importance of INCRs' incentivisation that eventually will lead to the expansion of resources, given their appropriate compensation/profit generation. We envision that the sufficient deployment of INCRs will be followed by the rapid development of the next generation of low latency applications, that currently cannot be supported by the typical "client-to-cloud" network model. Then in an context where INCRs and LLAs exist in abundance, potential research directions include but are not limited to the practical aspects of *i*) secure and efficient delivery of applications, involving the technically details of delivering verified application images to INCRs, *ii*) users' privacy, addressing the challenges of protecting users' sensitive data when connected to third-party INCRs, *iii*) safe compensation of INCRs, concerning the technical details of INCRs' compensation mechanisms, and *iv*) interoperability of different LLA delivery systems, that addresses the problem of users' seamless engagement and transition from one connectivity context to another.

Next we enumerate potential immediate research directions of the work presented in this thesis.

7.2.1 LLA Deployment under Disrupted Connectivity Conditions

Given the quality assessment framework, presented in Chapter 3, as a feasibility tool of crowdsourcing media delivery in urban railway networks, future work could investigate practical aspects of content delivery. To begin with, the quality assessment framework can be used at the core of a chunk scheduler, which decides who receives each chunk and when she shares it with the rest of the group, with the goal of optimising the delivery of the content. In fact, given the performance of a chunk scheduler the framework would be capable of recommending content according to the predicted quality of delivery, *e.g.,* if you join content A you will not experience any disruptions as apposed to content B which supported quality is poor. Such a chunk scheduler and quality delivery predictor can be combined to incorporate the monetary compensation of a user, *i.e.,* indirect incentives, for sharing successfully a chunk with the rest of the group. Finally, this work could be extended to include stochastic connectivity disruptions in more general settings, since in our current model the periods of disconnection are known, by also considering the assistance of the edge computing infrastructure [202, 203], *e.g.,* optimise the caching of content into edge computing resources along the journey of a group of commuters.

7.2.2 LLA Deployment under Mobile and Fixed Connectivity Conditions

Regarding LLA provisioning mechanisms, like the Edge-MAP and FogSpot as described in Chapters 4 and 5 respectively, future efforts should be directed in supporting diverse applications in an

unified way. As a first step, a rigorous study should be performed with respect to the conditions under which different provisioning approaches could be combined in a unified provisioning framework. After that, the research efforts should be focused on the problem of offering different types of VMs, at each market, with emphasis to INCRs' strategies for profit maximisation, when allocating their raw resources (like CPUs, storage, *etc.*) in different quantities for each VM. In that way, applications with different computing requirements could be supported. Another important aspect that requires investigation is related to multi-user applications, *e.g.,* cloud gaming, where the QoS gain of LLAs is affected by the overall experience of a group of users. Specifically, in the context of multi-user applications, new market mechanisms have to be developed for addressing the challenges of the joint bidding for resources in an efficient way. Lastly, in practical aspects of provisioning mechanisms, minimising the distribution overhead of application boot images, in a secure way, and implementing a realistic bidding mechanism for resources, by technologies that address trust issues like blockchain [179], create a promising research direction.

7.2.3 Complementary VNFs for LLAs

The deployment of VNFs for enhancing the LLAs performance over arbitrary topologies, as described in Chapter 6, is an interesting, pristine, and challenging problem that requires further investigation. In particular, frameworks developed with the purpose of handling virtual network functions, as response to LLA instances provisioning, should set as priority the fine-grained flow handling of LLA connections. The means of achieving such a goal in a cost-efficient way, *i.e.,* by occupying the least possible computing resources, require the design of a responsive framework that will react instantly to new LLA connections. Along these lines, an initial point could be the rigorous parametrisation of NFV nodes in the context of a regulated market, like in DRENCH, with the assistance of machine learning tools. Ideally, VNF management frameworks should be included as components of the LLAs' deployment market mechanisms.

Bibliography

[1] Anr connect project. [online]. available: http://anr-connect.org/.

[2] Bluetooth technology website. In *Bluetooth.com*.

[3] Cisco. cisco visual networking index: Global mobile data traffic forecast update, 2013-2018. white paper, [online] http://goo.gl/l77haj, 2014.

[4] Content centric networking project. [online]. available: http://www.ccnx.org/.

[5] Fp7 4ward project. [online]. available: http://www.4ward-project.eu/.

[6] Fp7 comet project.[online]. available:http://www.comet-project.org/.

[7] Fp7 convergence project. [online]. available: http://www.ict-convergence.eu/.

[8] Fp7 psirp project. [online]. available: http://www.psirp.org/.

[9] Fp7 pursuit project. [online]. available: http://www.fp7- pursuit.eu/pursuitweb/.

[10] Fp7 sail project. [online]. available: http://www.sail-project.eu/.

[11] ”market”. oxforddictionaries.com: Oxford university press.

[12] Nsf mobility first project. [online]. available: http://mobilityfirst.winlab.rutgers.edu/.

[13] Pplive: http://www.pplive.com.

[14] Smart insights survey: `http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics`.

[15] Tvunetworks: http://www.tvunetworks.com.

[16] ”wi-fi direct — wi-fi alliance”. In *Wi-fi.org*.

[17] Wifi alliance. wi-fi direct: http://www.wi-fi.org.

[18] G. Aceto, A. Botta, W. De Donato, and A. Pescapè. Cloud monitoring: A survey. *Computer Networks*, 57(9):2093–2115, 2013.

[19] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. Usage patterns in an urban wifi network. volume 18, pages 1359–1372. IEEE Press, 2010.

[20] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: Automated data placement for geo-distributed cloud services. 2010.

[21] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation*, 1(3):16, 2013.

[22] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[23] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*, pages 1–8. IEEE, 2016.

[24] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. 10:19–19, 2010.

[25] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

[26] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese, et al. Conga: Distributed congestion-aware load balancing for datacenters. 44(4):503–514, 2014.

[27] Amazon.com. Amazon elastic compute cloud. Online at `http://aws.amazon.com/ec2/`, 2009.

[28] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 286–298. ACM, 2007.

[29] T. Andersson, C. Andersson, and A. Talman. Sets in excess demand in simple ascending auctions with unit-demand bidders. *Annals of Operations Research*, 211(1):27–36, 2013.

[30] T. Andersson and A. Erlanson. Multi-item vickrey–english–dutch auctions. *Games and Economic Behavior*, 81:116–129, 2013.

[31] M. Arumaithurai, J. Chen, E. Monticelli, X. Fu, and K. K. Ramakrishnan. Exploiting icn for flexible management of software-defined networks. In *Proceedings of the 1st international conference on Information-centric networking*, pages 107–116. ACM, 2014.

[32] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou. On uncoordinated service placement in edge-clouds. In *Cloud Computing Technology and Science (CloudCom), 2017 IEEE International Conference on*, pages 41–48. IEEE, 2017.

[33] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[34] M. Avgerinou, P. Bertoldi, and L. Castellazzi. Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency. *Energies*, 10(10):1470, 2017.

[35] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. A quest for an internet video quality-of-experience metric. In *Proceedings of the 11th ACM workshop on hot topics in networks*, pages 97–102. ACM, 2012.

[36] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 343–352. ACM, 2004.

[37] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb. Simplifying cyber foraging for mobile devices. In *Mobile systems, applications and services*. ACM, 2007.

[38] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *INFOCOM, 2013 Proceedings IEEE*, pages 1285–1293. IEEE, 2013.

[39] E. Bastug, M. Bennis, and M. Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89, 2014.

[40] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini. Convergence of manet and wsn in iot urban scenarios. *IEEE Sensors Journal*, 13(10):3558–3567, 2013.

[41] T. Benson, A. Anand, A. Akella, and M. Zhang. Microte: Fine grained traffic engineering for data centers. page 8, 2011.

[42] S. Bhardwaj, L. Jain, and S. Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[43] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, and D. H. Lorenz. Guaranteeing high availability goals for virtual machine placement. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 700–709. IEEE, 2011.

[44] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera. A stable network-aware vm placement for cloud systems. In *Proceedings of the 2012 12th IEEE/ACM*

International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pages 498–506. IEEE Computer Society, 2012.

[45] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson. Characterizing, modeling, and generating workload spikes for stateful services. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 241–252. ACM, 2010.

[46] C. Boldrini, M. Conti, and A. Passarella. Exploiting users social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing*, 4(5):633–657, 2008.

[47] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. 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. ACM, 2012.

[48] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem. Analysis of end-to-end delay measurements in internet. In *Proc. of the Passive and Active Measurement Workshop-PAM*, volume 2002. sn, 2002.

[49] P. Buxmann, T. Hess, and S. Lehmann. Software as a service. *Wirtschaftsinformatik*, 50(6):500–503, 2008.

[50] M. Cardosa, M. R. Korupolu, and A. Singh. Shares and utilities based power consolidation in virtualized server environments. In *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*, pages 327–334. IEEE, 2009.

[51] B. Carpenter and S. Brim. Middleboxes: Taxonomy and issues. Technical report, 2002.

[52] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 918–928. IEEE, 2004.

[53] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *Developing an Infrastructure for Mobile and Wireless Systems*, pages 59–68. Springer, 2002.

[54] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 163–174. ACM, 2003.

[55] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod. Streaming mobile augmented reality on mobile phones. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 181–182. IEEE, 2009.

[56] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, pages 22–24, 2012.

[57] J. Choi, J. Han, E. Cho, T. T. Kwon, and Y. Choi. A survey on content-oriented networking for efficient content delivery. *Communications Magazine, IEEE*, 49(3):121–127, 2011.

[58] N. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[59] S. Choy, B. Wong, G. Simon, and C. Rosenberg. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proceedings of the 11th annual workshop on network and systems support for games*, page 2. IEEE Press, 2012.

[60] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[61] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[62] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[63] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., 1996.

[64] R. Cohen and G. Nakibly. On the computational complexity and effectiveness of n-hub shortest-path routing. volume 16, pages 691–704. IEEE Press, 2008.

[65] I. Corp. Ibm tivoli provisioning manager, 2002.

[66] C. Courcoubetis and R. Weber. *Pricing communication networks: economics, technology and modelling*. John Wiley & Sons, 2003.

[67] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[68] R. Das, S. Yarlanki, H. Hamann, J. O. Kephart, and V. Lopez. A unified approach to coordinated energy-management in data centers. In *Proceedings of the 7th International Conference*

on Network and Services Management, pages 504–508. International Federation for Information Processing, 2011.

[69] P. De and S. Roy. Vmspreader: multi-tier application resiliency through virtual machine striping. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 185–192. IEEE, 2011.

[70] J. DiGiglio and D. Ricci. High performance, open standard virtualization with nfv and sdn. *Wind River*, 2013.

[71] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. volume 41, pages 29–40. ACM, 2013.

[72] T. Dörnemann, E. Juhnke, and B. Freisleben. On-demand resource provisioning for bpel workflows using amazon's elastic compute cloud. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 140–147. IEEE, 2009.

[73] N. Eagle. txteagle: Mobile crowdsourcing. pages 447–456, 2009.

[74] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs. Resource allocation for distributed cloud: concepts and research challenges. *IEEE network*, 25(4), 2011.

[75] E. T. S. I. (ETSI). Network function virtualization. *http://www.etsi.org/technologies-clusters/technologies/nfv*, 2013.

[76] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *Symposium on Theory of Computing*. ACM, 2004.

[77] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM, 2003.

[78] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In *NSDI*, volume 14, pages 533–546, 2014.

[79] P. Ferguson and G. Huston. *Quality of service: delivering QoS on the Internet and in corporate networks*, volume 1. Wiley New York, 1998.

[80] G. P. Fettweis. The tactile internet: Applications and challenges. *IEEE Vehicular Technology Magazine*, 9(1):64–70, 2014.

[81] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *ACM SIGCOMM Computer Communication Review*, 30(4):43–56, 2000.

[82] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[83] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[84] M. J. Freedman, M. Arye, P. Gopalan, S. Y. Ko, E. Nordstrom, J. Rexford, and D. Shue. Service-centric networking with scaffold. Technical report, DTIC Document, 2010.

[85] D. F. Galletta, R. Henry, S. McCoy, and P. Polak. Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, 5(1):1, 2004.

[86] R. Gandhi, H. H. Liu, Y. C. Hu, G. Lu, J. Padhye, L. Yuan, and M. Zhang. Duet: Cloud scale load balancing with hardware and software. volume 44, pages 27–38. ACM, 2015.

[87] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella. Stratos: A network-aware orchestration layer for virtual middleboxes in clouds. *arXiv preprint arXiv:1305.0209*, 2013.

[88] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, pages 163–174. ACM, 2014.

[89] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *USITS*, volume 1, pages 4–4, 2001.

[90] R. Gupta, S. K. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti. A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints. In *Services Computing, 2008. SCC'08. IEEE International Conference on*, volume 2, pages 39–46. IEEE, 2008.

[91] B. Han, P. Hui, V. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan. Cellular traffic offloading through opportunistic communications: a case study. In *Proceedings of the 5th ACM workshop on Challenged networks*, pages 31–38. ACM, 2010.

[92] M. R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 51–60. ACM, 2009.

[93] C. Höfer and G. Karagiannis. Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, 2(2):81–94, 2011.

[94] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589, 2011.

[95] K. Z. Ibrahim, S. Hofmeyr, C. Iancu, and E. Roman. Optimized pre-copy live migration for memory intensive applications. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 40. ACM, 2011.

[96] C. V. N. Index. Global mobile data traffic forecast update, 2015–2020 white paper. *link: http://goo. gl/ylTuVx*, 2016.

[97] S. Ioannidis, A. Chaintreau, and L. Massoulie. Optimal and scalable distribution of content updates over a mobile social network. In *INFOCOM 2009, IEEE*, pages 1422–1430. IEEE, 2009.

[98] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879, 2011.

[99] H. Izakian, A. Abraham, and B. T. Ladani. An auction method for resource allocation in computational grids. *Future Generation Computer Systems*, 26(2):228–235, 2010.

[100] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne. Rtp: A transport protocol for real-time applications. 2003.

[101] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible. Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 72–79. IEEE, 2011.

[102] H. Jiang and C. Dovrolis. Why is the internet traffic bursty in short time scales? In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 241–252. ACM, 2005.

[103] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. Joint vm placement and routing for data center traffic engineering. In *INFOCOM, 2012 Proceedings IEEE*, pages 2876–2880. IEEE, 2012.

[104] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. *IEEE Transactions on Services Computing*, 9(6):895–909, 2015.

[105] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. *ACM SIGCOMM Computer Communication Review*, 38(4):51–62, 2008.

[106] G. Jung, K. R. Joshi, M. Hiltunen, R. D. Schlichting, C. Pu, et al. Generating adaptation policies for multi-tier applications in consolidated server environments. In *Autonomic Computing, 2008. ICAC'08. International Conference on*, pages 23–32. IEEE, 2008.

[107] S. Jung, U. Lee, A. Chang, D.-K. Cho, and M. Gerla. Bluetorrent: Cooperative content sharing for bluetooth users. *Pervasive and Mobile Computing*, 3(6):609–634, 2007.

[108] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. 35(4):253–264, 2005.

[109] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 202–208. ACM, 2009.

[110] K. V. Katsaros, B. Yang, W. K. Chai, and G. Pavlou. Low latency communication infrastructure for synchrophasor applications in distribution networks. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 392–397. IEEE, 2014.

[111] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford. Hula: Scalable load balancing using programmable data planes. page 10, 2016.

[112] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou. Microcast: Cooperative video streaming on smartphones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 57–70. ACM, 2012.

[113] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[114] A. H. Khalaj, T. Scherer, and S. K. Halgamuge. Energy, environmental and economical saving potential of data centers with various economizers across australia. *Applied energy*, 183:1528–1549, 2016.

[115] L. Kleinrock. *Queueing systems, volume 2: Computer applications*. Wiley New York, 1976.

[116] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. 37(4):181–192, 2007.

[117] R. Landa, M. Charalambides, R. G. Clegg, D. Griffin, and M. Rio. Self-tuning service provisioning for decentralized cloud applications. *IEEE Transactions on Network and Service Management*, 13(2):197–211, 2016.

[118] J. LeBrun and C.-N. Chuah. Bluetooth content distribution stations on public transit. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 63–65. ACM, 2006.

[119] I. Lee and K. Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.

[120] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

[121] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 17–24. IEEE, 2009.

[122] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi. A measurement study on tcp behaviors in hspa+ networks on high-speed rails. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2731–2739. IEEE, 2015.

[123] V. Liu, D. Halperin, A. Krishnamurthy, and T. E. Anderson. F10: A fault-tolerant engineered network. In *NSDI*, pages 399–412, 2013.

[124] G. Lyons and K. Chatterjee. A human perspective on the daily commute: Costs, benefits and trade-offs. *Transport Reviews*, 28.

[125] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, D. J. Scott, R. Mortier, A. Chaudhry, B. Singh, J. Ludlam, et al. Jitsu: Just-in-time summoning of unikernels. In *NSDI*, pages 559–573, 2015.

[126] A. Madhavapeddy and D. J. Scott. Unikernels: the rise of the virtual library operating system. *Communications of the ACM*, 57(1):61–69, 2014.

[127] L. Mai, L. Rupprecht, A. Alim, P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf. Netagg: Using middleboxes for application-specific on-path aggregation in data centres. pages 249–262, 2014.

[128] N. McKeown. Software-defined networking. *INFOCOM keynote talk*, 17(2):30–32, 2009.

[129] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 58–69. ACM, 2008.

[130] P. Mell and T. Grance. The nist definition of cloud computing. *Communications of the ACM*, 53(6):50, 2010.

[131] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[132] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[133] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. Internet denial of service: Attack and defense mechanisms (radia perlman computer networking and security). 2004.

[134] D. Mishra and D. C. Parkes. Multi-item vickrey–dutch auctions. *Games and Economic Behavior*, 66(1):326–347, 2009.

[135] D. Mishra and D. Talman. Characterization of the walrasian equilibria of the assignment model. *Journal of Mathematical Economics*, 46(1):6–20, 2010.

[136] P. Mockapetris and K. J. Dunlap. *Development of the domain name system*, volume 18. ACM, 1988.

[137] N. Nasser, A. Hasswa, and H. Hassanein. Handoffs in fourth generation heterogeneous networks. *IEEE Communications Magazine*, 44(10):96–103, 2006.

[138] J. Ni and X. Bai. A review of air conditioning energy performance in data centers. *Renewable and sustainable energy reviews*, 67:625–640, 2017.

[139] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman. Serval: An end-host stack for service-centric networking. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 7–7. USENIX Association, 2012.

[140] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler. Standardized protocol stack for the internet of (important) things. *IEEE communications surveys & tutorials*, 15(3):1389–1406, 2013.

[141] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: a framework for nfv applications. pages 121–136, 2015.

[142] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder. Heuristics for vector bin packing. *research. microsoft. com*, 2011.

[143] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh. A cyber-physical systems approach to energy management in data centers. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 168–177. ACM, 2010.

[144] R. Pastor-Satorras and A. Vespignani. *Evolution and structure of the Internet: A statistical physics approach*. Cambridge University Press, 2007.

[145] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu, et al. Ananta: Cloud scale load balancing. volume 43, pages 207–218. ACM, 2013.

[146] A.-M. K. Pathan and R. Buyya. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 4, 2007.

[147] E. Paulos and E. Goodman. The familiar stranger: anxiety, comfort, and play in public places. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 223–230. ACM, 2004.

[148] M. Peng, S. Yan, K. Zhang, and C. Wang. Fog-computing-based radio access networks: issues and challenges. *IEEE Network*, 30(4):46–53, 2016.

[149] M.-A. Pierre-Louis, C. J. Paul, and S. Radhakrishnan. Method and apparatus for managing boot images in a distributed data processing system, 2002. US Patent 6,421,777.

[150] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[151] I. Psaras, S. Reñé, K. Katsaro, V. Sourlas, G. Pavlou, N. Bezirgiannidis, S. Diamantopoulos, I. Komnios, and V. Tsaoussidis. Keyword-based mobile application sharing. In *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, pages 1–6. ACM, 2016.

[152] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simple-fying middlebox policy enforcement using sdn. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 27–38. ACM, 2013.

[153] S. Rajagopalan, D. Williams, and H. Jamjoom. Pico replication: A high availability framework for middleboxes. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 1. ACM, 2013.

[154] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield. Split/merge: System support for elastic execution in virtual middleboxes. In *NSDI*, volume 13, pages 227–240, 2013.

[155] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. Mar: A commuter router infrastructure for the mobile internet. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 217–230. ACM, 2004.

[156] H. Rong, H. Zhang, S. Xiao, C. Li, and C. Hu. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691, 2016.

[157] R. W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 1973.

[158] A. E. Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[159] J. Rulifson. Decode encode language. *RFC5*, 1969.

[160] L. Saino, I. Psaras, and G. Pavlou. Icarus: a caching simulator for information centric networking (icn). In *Proceedings of the 7th International ICST conference on Simulation Tools and Techniques*, pages 66–75. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[161] Z. Sanaei, S. Abolfazli, A. Gani, and M. Shiraz. Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In *Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on*, pages 14–19. IEEE, 2012.

[162] J. K. Sankaran. On a dynamic auction mechanism for a bilateral assignment problem. *Mathematical Social Sciences*, 28(2):143–150, 1994.

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

[164] H. Seferoglu, L. Keller, B. Cici, A. Le, and A. Markopoulou. Cooperative video streaming on smartphones. In *49th Annual Allerton Conference*.

[165] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 24–24. USENIX Association, 2012.

[166] S. Shanbhag, N. Schwan, I. Rimac, and M. Varvello. Soccer: Services over content-centric routing. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 62–67. ACM, 2011.

[167] E. B. Shapiro. Network timetable. *RFC4*, 1969.

[168] L. S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of game theory*, 1(1):111–130, 1971.

[169] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya. Pricing cloud compute commodities: A novel financial economic model. In *Proceedings of the 2012 12th*

IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pages 451–457. IEEE Computer Society, 2012.

[170] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh. A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 559–570. IEEE, 2011.

[171] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui. Cloud gaming: architecture and performance. *Ieee Network*, 27(4):16–21, 2013.

[172] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24, 2012.

[173] L. Shi, B. Butler, D. Botvich, and B. Jennings. Provisioning of requests for virtual machine sets with placement constraints in iaas clouds. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 499–505. IEEE, 2013.

[174] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):71–83, 2014.

[175] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *INFOCOM, 2011 Proceedings IEEE*, pages 66–70. IEEE, 2011.

[176] S.Kumar, M.tufail, S.Maji, C.captari, and S.Homma. Service Function Chaining Use Cases In Data Centers. *IETF draft*, 2016.

[177] M. Stiemerling and S. Kiesel. A system for peer-to-peer video streaming in resource constrained mobile environments. In *Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities*, pages 25–30. ACM, 2009.

[178] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 73–86. ACM, 2002.

[179] M. Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[180] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas. Bits and coins: Supporting collaborative consumption of mobile internet. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2146–2154. IEEE, 2015.

[181] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData & SocialInformatics 2015*, page 28. ACM, 2015.

[182] C.-L. Tsao and R. Sivakumar. On effectively exploiting multiple wireless interfaces in mobile hosts. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 337–348. ACM, 2009.

[183] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2014.

[184] M. Venkataraman and M. Chatterjee. Inferring video qoe in real time. *IEEE network*, 25(1), 2011.

[185] D. C. Verma. *Content distribution networks: an engineering approach*. John Wiley & Sons, 2003.

[186] W. Voorsluys, J. Broberg, and R. Buyya. Introduction to cloud computing. *Cloud computing: Principles and paradigms*, pages 1–41, 2011.

[187] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the web from dns. 4:17–17, 2004.

[188] M. Walfish, J. Stribling, M. N. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *OSDI*, volume 4, pages 15–15, 2004.

[189] F. Wang, J. Liu, and Y. Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1364–1372. IEEE, 2008.

[190] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on selected areas in communications*, 14(7):1228–1234, 1996.

[191] Z. Wang, N. Tolia, and C. Bash. Opportunities and challenges to unify workload, power, and cooling management in data centers. In *Proceedings of the Fifth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, pages 1–6. ACM, 2010.

[192] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan. Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–10. IEEE, 2011.

[193] A. Whitmore, A. Agarwal, and L. Da Xu. The internet of things a survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274, 2015.

[194] D. Wilcox, A. McNabb, and K. Seppi. Solving virtual machine packing with a reordering grouping genetic algorithm. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 362–369. IEEE, 2011.

[195] C. Williamson, E. Halepovic, H. Sun, and Y. Wu. Characterization of cdma2000 cellular data network traffic. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages Z000–719. IEEE, 2005.

[196] T. Wood, K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang. Toward a software-based network: integrating software defined networking and network function virtualization. *IEEE Network*, 29(3):36–41, 2015.

[197] W. Wu, R. T. Ma, and J. C. Lui. Distributed caching via rewarding: An incentive scheme design in p2p-vod systems. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):612–621, 2014.

[198] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs. Network function placement for nfv chaining in packet/optical datacenters. *Journal of Lightwave Technology*, 33(8):1565–1570, 2015.

[199] H. Xu and B. Li. Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing*, 1(2):158–171, 2013.

[200] J. Xu and J. Fortes. A multi-objective approach to virtual machine management in datacenters. In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 225–234. ACM, 2011.

[201] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, et al. A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE*, 16(2):1024–1049, 2014.

[202] B. Yang, W. K. Chai, G. Pavlou, and K. V. Katsaros. Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on*, pages 136–141. IEEE, 2016.

[203] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou. Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications. *IEEE Transactions on Network and Service Management*, 2018.

[204] B. Yang, Z. Xu, W. K. Chai, W. Liang, D. Tuncer, A. Galis, and G. Pavlou. Algorithms for fault-tolerant placement of stateful virtualized network functions. 2018.

[205] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan. A framework for partitioning and execution of data stream applications in mobile cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 40(4):23–32, 2013.

[206] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[207] B. Zhang, N. Mor, J. Kolb, D. S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. A. Lee, and J. Kubiatowicz. The cloud is not enough: Saving iot from the cloud. In *HotStorage*, 2015.

[208] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[209] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[210] W. Zhao, D. Olshefski, and H. Schulzrinne. Internet quality of service: An overview. *Columbia University, New York, New York, Technical Report CUCS-003-00*, 2000.

[211] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu. Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter. *Computer networks*, 67:141–153, 2014.