Performance modelling and simulation of the Mobile Cloud Network

Tärneberg, William

2015

Document Version:
Publisher's PDF, also known as Version of record

Performance modelling and simulation of the Mobile Cloud Network

–

William Tärneberg

Lund 2015

Department of Electrical and Information Technology
Lund University
Box 118, SE-221 00 LUND
SWEDEN

This thesis is set in Computer Modern 10pt
with the LaTeX Documentation System

Series of licentiate and doctoral theses
No. 79
ISSN 1654-790X
ISBN 978-91-7623-580-5

Printed in Sweden by *Tryckeriet i E-huset*, Lund.
November 2015.

Abstract

The Mobile Cloud Network is an emerging distributed cloud infrastructure paradigm that attempts to accommodate the evolution of application's execution paradigms and how content is distributed. The Mobile Cloud Network employs a distributed cloud infrastructure with data centres of varying capacity, embedded in the core and access networks. Resources are thinned over the network and arguably decay in capacity towards the capillaries of the network. For an operator of such an infrastructure, cost and infrastructure integrity is their primary concern. Although these aspirations seem trivial, achieving them in a highly dynamic and heterogeneous distributed infrastructure, successfully at scale is non trivial. In this work, we have focused on modelling the dynamics of the Mobile Cloud Network infrastructure and what methods the operator of a Mobile Cloud Network can employ to minimise the cost of operating the infrastructure. The reduced parameter model, reveals the non-linear nature of the Mobile Cloud Network. The system's performance is evaluated using a cost function that encompasses the cost of executing the applications, the cost of the incurred link usage, and the expected performance of the application. Furthermore, simulations reveal that the proposed methods can achieve near optimal placement at reduced time-complexity.

Preface

This licentiate thesis concludes my work as a licentiate candidate and is comprised of two parts. The first part gives an overview of the research field in which I have been working during my licentiate and a brief summary of my contribution in it. The second part is composed of four included papers that constitute my main scientific work:

[1] William Tärneberg, Maria Kihl, "Workload displacement and mobility in an omnipresent cloud topology" Presented at *IEEE SoftCOM 2014*, Split, Croatia, Sept 2014.

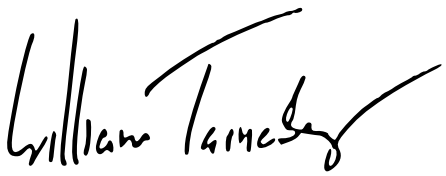
[2] Jakub Krzywda, William Tärneberg, Per-Olov Östberg, Maria Kihl, Erik Elmroth "Telco Clouds: Modelling and Simulation" Presented at *Closer 2015*, Lisbon, Portugal, May 2015.

[3] William Tärneberg, Amardeep Mehta, Johan Tordsson, Maria Kihl, Erik Elmroth, "Resource Management Challenges for the Telco cloud" Presented at *Feedback Computing Workshop at CPSWeek 2015. 2015)*, Seattle, United States, April, 2015.

[4] William Tärneberg, Amardeep Mehta, Eddie Wadbro, Johan Tordsson, Johan Eker, Maria Kihl, Erik Elmroth, "Dynamic Application Placement in the Mobile Cloud Network" Work in progress.

Acknowledgements

This work is funded in part by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. I am also a member of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council (VR) and the Excellence Center Linköping - Lund in Information Technology (ELLIIT). In addition, I am also funded by the Mobile and Pervasive Computing Institute Lund University (MAPCI).



William Tärneberg

List of Acronyms and Abbreviations

DC Data Centre 3

MNO Mobile Network Operator 84

VM Virtual Machine 10

PM Physical Machine 15

MD Mobile Devices 51

RBS Radio Base Station 4

WAN Wide Area Network 15

RAN Radio Access Network 51

MAN Metropolitan Area Network 54

QoS Quality of Service 17

IP Infrastructure Provider 51

SLO Service Level Objective 8

SLA Service Level Agreement 79

IaaS Infrastructure as a Service 60

SaaS Software as a Service 60

RAN Radio Access Network 51

MCN Mobile Cloud Network 4

IoT Internet of Things 82

IoE Internet of Everything 4

WFS Weighted Fair Share 17

PC Personal Computer 3

OPEX Operational Expenditure 84

MIP Mixed Integer Programming 117

RTT Round Trip Time 8

CDN Content Distribution Network 7

TSP Telecom Service Provider 6

RTD Round Trip Delay 7

MPC Model Predictive Control 17

Contents

Abstract iii

Preface v

Acknowledgements vii

List of Acronyms and Abbreviations ix

Contents xiii

I Overview of Research Field 1

1 Introduction 3

1.1 Cloud computing 5

2 Resource management challenges in a Mobile Cloud Network 7

2.1 Service offering paradigm 8

2.2 Management objectives 8

2.3 Resource management dynamics 8

3 Summary and Contributions 11

3.1 Research contributions 11

3.2 General Conclusions and Future Work 13

4 Non-intrusive replication for fault tolerance 15

4.1 System 15

4.2 Challenge 16

4.3 Model and system dynamics 17

4.4 Approach 18
4.5 Preliminary evaluation 19
4.6 Future work 19

References **20**

II Included Papers **27**

Workload displacement and mobility in an omnipresent cloud topology **31**
1 Introduction 33
2 Omnipresent cloud 35
3 Targeted scenario 36
4 Simulation model 36
5 Experiments 38
6 Results and discussions 39
7 Conclusions 44
References 44

Telco Clouds: Modelling and Simulation **49**
1 Introduction 51
2 The Telco cloud Architecture 51
3 Simulation challenges 55
4 Telco cloud Dynamics 56
5 Existing models 57
6 Telco cloud Meta-model 62
7 Simulation showcase 67
8 Conclusions 70
References 71

Resource Management Challenges for the Infinite Cloud **79**
1 Introduction 81
2 Infinite cloud motivation 81
3 Challenges 82
4 Approach 84
5 Acknowledgements 85

References 85

Dynamic Application Placement in the Mobile Cloud Network **91**

1 Introduction 93
2 Resource Management Challenges 94
3 System model 97
4 Optimisation Formulation 101
5 Application Placement Methods 104
6 Evaluation 106
7 Related work 115
8 Conclusions 118
9 Acknowledgements 118
References 118

Part I

Overview of Research Field

Chapter 1

Introduction

The promise of computing as a utility was first coined in the 1960s when John McArthy predicted its eventual arrival [1]. At the time, computing resources were often shared mainframes, under the paradigm "one computer, many users", but were not organised as a ubiquitous utility. In this era, the expense of personal computing outweighed the expense of communication, availability, and performance. Over the subsequent years, the Personal Computer (PC) matured and found itself useful in a "one user, one computer" world where content is cost-effectively consumed and produced locally as a result of relatively high communication costs and cheap hardware. Furthermore, often credited to [2], the concept of shared distributed computing resurfaced in the 1990s under the term grid computing. Grid computing departed from the prevailing supercomputer paradigm at the time by proposing scalable computing infrastructures, employing commodity hardware, universally accessible, and distributed over several geographically separated clusters. Additionally, with its origins in compute resource scavenging, such as SETI [3], the grid-computing era advanced the notion of parallel and distributed computing. Furthermore, the term cloud computing came into fashion in the mid 2000s [4] when resource virtualisation came to fruition and enabled infrastructure provides to offer a universal and transferable compute resource. Services such as the AWS [5], Google Cloud, Microsoft Azure [6], and Rackspace emerged out of this era and set the direction for large scale public clouds. The technology and deployment paradigms they trailblazed are now enabling smaller vendors, corporations, and private individuals to successfully construct their own clouds, at a competitive cost. Although we are progressively approaching a compute utility resource offering, the service paradigm is still point-to-point where the applications are executed in centralised Data Centres (DCs) that distribute content centrally.

With a continuing drop in the cost of communication, large scale distributed ap-

plications such as Internet of Everything (IoE) services are beginning to emerge. With IoE, where everything is connected with everything and everyone, creation and consumption is by definition, distributed. Applications do not necessarily have a clear origin, and its relevance might be geographically bounded. In addition, the amount of content and data generated at any given point in space is exploding, subjecting the communication infrastructure to new capacity challenges. Similarly, content consumption is also becoming more decentralised, ranging from local to global reach. This form of computing, "many computers, one user" where everything everywhere is connected, is referred to as pervasive computing [7].

To accommodate the increased degree of distribution and increased network traffic, cloud compute resources are now gradually being decentralised and dispersed into the core and access networks where more and more data and content is consumed and generated. The compute capacity can for example reside in so called edge data centres, proposedly in the Radio Base Stations (RBSs), at switching stations, or in peoples homes. A distributed topology enables applications and their resulting traffic streams to be contained to where they are relevant, incur less global contention, and to where they can most cost effectively achieve its desired performance. This infrastructural paradigm is in this thesis referred to as the Mobile Cloud Network (MCN).

In contrast to the currently prevailing centralised cloud infrastructure, the MCN employs a distributed resource- and cost-heterogeneous infrastructure embedded into the core and access networks [8]. Managing the resources of the multidimensional MCN is non-trivial. To minimise operational cost, resources need to be allocated and applications placed where they also are able to meet their performance objectives, given communication latency, and the spatial origin of their demand. In this thesis we primarily address this specific resource management challenges. To study the properties of, and formulate resource management principals for, such a system I developed;

- Performance models encompassing; applications, demand, and infrastructure.
- Simulation frameworks capturing the behaviour of the MCN
- Resource management principals for the MCN centring around heuristic cost optimal application placement.

Complimentary to our work on resource managing the MCN we are also exploring how to achieve non-intrusive replication between DCs. Non-intrusive in the sense that the replication traffic does not interfere with the service traffic.

This thesis is structured as follows; Section 1.1 provides a brief overview of cloud computing. Sections 1.1.1 and 1.1.2 highlight the main paradigm shift between a centralised and a distributed cloud. Chapter 2 introduces some of the key challenges addressed by this thesis. Furthermore, Chapter 3 covers the four included papers and what my contributions are in each of them. The chapter ends with Section 3.2 which

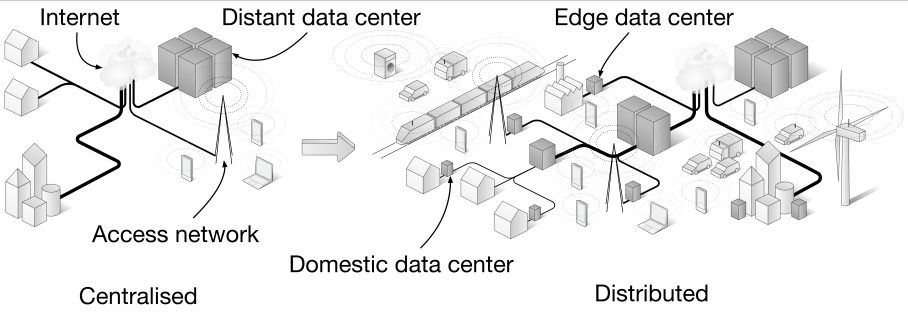


Figure 1.1: Mobile Cloud Network

covers how the content of this thesis ties in with our ongoing and planned work. Lastly, Chapter 4 introduces my ongoing work on non-intrusive replication for fault tolerance.

1.1 Cloud computing

Cloud computing has enabled application developers to tap into a reservoir of ubiquitous computing resources. Modern applications are more often than not deployed and executed in one or across multiple public clouds. The deployment paradigm enables application owners to continuously scale their deployed applications to their demand, using only the resources they need, at a competitive cost. The currently available cloud capacity conventionally originates from large DCs. These DCs are increasingly not only used to provide commodity computing capacity, but also storage, security, software platforms, and networking.

1.1.1 Centralised clouds

Much of today's public cloud infrastructure is centralised to large DCs, see Figure 1.1. The operators of such infrastructures achieve economies of scale by constructing large clusters using commodity hardware, being built on cheap land at locations where energy comes at a low cost and is locally produced, and by being holistically managed [9]. A cloud provider conventionally serves a continent with a handful of large DCs. The geographic separation been the DC and the end-user introduces an unwanted performance-inhibiting communication latency [10]. The intra-continental communication latencies between DCs and their clients are or average in the tens of *ms* range while the inter-continental communication latencies are above a hundred *ms* [11].

1.1.2 Distributed clouds and Telcos

The above-mentioned MCN infrastructure paradigm is still in its infancy and is by no means technically defined by any set of standards. The definition of the MCN varies from a Telecom-centric distributed cloud infrastructure that solely hosts virtualised Telecom services and infrastructure [12, 13, 14], to a federated infrastructure, open to all and encompasses any available resources shared amongst the peers [15, 16, 17] in the network. Correspondingly, the paradigm has just as many names, but is commonly referred to as either Telco-cloud, Infinite-cloud, Omnipresent-cloud, or Mobile Cloud Network.

The topology of future generation networks and the evolution of compute capacity dispersion is unknown. We therefore study the problems addressed below from a general perspective. In this thesis we approach the MCN in the middle-ground between the two extremes where cloud capacity is heterogeneously distributed throughout a Telecom-network, accessible by all, and managed concurrently with the network by the operator of that network, traditionally a Telecom Service Provider (TSP), see Figure 1.1.

Chapter 2

Resource management challenges in a Mobile Cloud Network

Although little work has been done on resource management of an MCN, optimal placement of content or web-sever replicas in distributed infrastructures or applications inside a DC is not a novel pursuit. Some of the application placement challenges facing the MCN are also found in Content Distribution Networks (CDNs). Centralised and distributed approaches have been developed for specific scenarios [18, 19]. Additionally, work has also been performed on how and when to migrate applications between DCs [20, 21, 22, 23, 24]. Work performed in [25] extend on previous work by proposing a topology-aware replica placement policy, but does not address user mobility and the resulting rate of change it introduces. Research presented in [23] does consider individual user location when migrating and duplicating user instances on a planetary scale, with the objective to reduce Round Trip Delay (RTD). Their approach does not take into account topology nor rapid changes in demand and its location. The literature does not address the holistic management of all of the system's heterogeneous resources at scale and does not take into account demand mobility. Additionally, the body of work lacks a system model.

This chapter introduces and discusses some of the primary holistic resource management challenges found in an MCN and how they have been addressed in the four included papers. Sections 2.1 and 2.2 elaborate on what services are offered and how they are reflected in the internal management of the MCN. Section 2.3 details an MCN's system dynamics and how they interplay with the system's management objectives.

2.1 Service offering paradigm

In our work we assume that applications are submitted by their owners to an MCN operated by a TSP, on the basis that the TSP hosts a significant enough demand for its applications. The application owner is assumed to be agnostic to the scale, capacity, and management objectives of the MCN. When submitting an application, its owner provides the TSP with a set of Service Level Objectives (SLOs), primarily concerning Round Trip Time (RTT) and resource commitments. Paper IV presents a model for a service or application, its constituent components and performance requirements.

2.2 Management objectives

The TSP is responsible for running the submitted applications in its MCN and is incited to do so in a manner that complies with its internal management agenda while meeting the SLO liaised with the application's owners. We make the assumption that it is the TSP's management objective to minimise its operational expenditure. Papers II and IV contribute with such cost models. The expenditure is determined by the sum of the cost of running the applications in the DCs and the incurred cost of the resulting network utilisation. Furthermore, when managing its infrastructure, the TSP has three primary degrees of freedom.which applications to admit, which pieces of infrastructure to run and when, and where to run the application components.

2.3 Resource management dynamics

The key resource management challenge addressed in this thesis is how to place and scale applications and their components in the MCN infrastructure. Placing heterogeneous application components, driven by time and spatially variant demand in a cost and capacity heterogeneous MCN is not trivial. Fundamentally, the TSP will seek to minimise its running cost by placing applications where they, given the prevailing demand constellation, incur the least amount of cost, while meeting the application's SLOs. Some of the challenges introduce by the above-mentioned dynamics are discussed below.

2.3.1 Application demand

Any resource management approach needs to continuously match a highly dynamic demand with a dynamic set of resources characterised by heterogeneous cost and capacity, while minimising the operational cost for the operator. The demand is time-variant in both space and quantity and can range from being very local to uniformly over a large geographic area. At two extremes, an offloaded mobile application might

only be consumed by one device in the network, while an IoE application will constitute a large number of users across a large section of the network. The performance overhead incurred by user and demand mobility are explored in Paper I and II. Furthermore, different demand distributions are studied in Paper IV.

2.3.2 Application structure and performance requirements

Modern applications are now more often than not split up into multiple components [26]. The various components that constitute an application have intermediary latency and throughput constrains, and independently have affinity and anti-affinity constrains to other application components in the infrastructure [27]. A resource usage model for multi-tier applications and constraints for affinity and anti-affinity are introduced in Paper IV.

2.3.3 Resource consumption and cost

Demand for a resident application incurs resource usage in both the DCs and the intermediate links. Each of these resources are subject to a unit/time operational cost. The unit/time cost of a compute resource in a DC varies across the network and is assumed to decrease with size, as the economies of scale diminish. Additionally, a proportion of the available capacity might be leased and therefore come at a cost premium. When this is the case, the operator is incited to down-prioritize the usage of those resources unless it avoids violating a performance commitment or constraint, such as an SLO. Applications consume an aggregate amount of resources proportional to its demand and the number of replicas over which it resides. A resource usage model for applications is introduced in Paper II. Additionally, different cost models are evaluated in Papers II and IV.

2.3.4 Evaluation domain

As demand and resource availability shifts, the system might find itself in a state where resources are being starved, costs are escalating, and application performance constraints are being violating. At this point, the system either needs to expel applications, add resources, or re-evaluate how the applications are placed given the new set of circumstances. The former two are not likely to be achieved quickly enough to bring the system to a desired state, in runtime. That is not to say that admission control might be practised at either the demand- or application submission- end, if the aggregate system becomes under-provisioned. Nevertheless, with either of the above remedies, the non-trivial challenge of selecting which applications to expel and which resources to provision remains. In this thesis we do not take into account admission

control, but rather focus on managing the applications the system has already committed to.

In the traditional cloud service offering paradigm, cloud providers are practising resource throttling or other forms of prioritisation based on client class levels as a tool to manage its resources [28]. However, in a cost and resource heterogeneous distributed system such as an MCN, only minding a subset of applications regardless of what they contribute to the state of the system will not ensure that the system will converge to a desired state. Nevertheless, in the ideal case, re-evaluating all applications across all nodes ensures that costs are minimised, given the prevailing workload. However, continuously re-evaluating all applications over all possible resources at the system's aggregate rate of change is not scalable. Which applications to re-evaluate, over which resources, and when is one of the key challenges to achieving a scalable resource management solution.

The number of placement options grow exponentially with the number of DCs and applications. The sheer number of potential application components and compute resources in the network makes it impractical to re-evaluate the placement of each application at system's rate of change. We explore this limitation in Paper IV, where we observe the modelling challenge that lies in constructing a cost function that is able to capture the dynamics of the system and its individual components so that you can determine when, which resources, and which applications to re-evaluate to achieve the above resource management objectives. The scalability challenge is discussed more generally in Paper III.

2.3.5 Migration

When an application or a set of its components is replicated or moved as a consequence of a management decision, the state of the application and the Virtual Machine (VM) on which it runs needs to be replicated to and started at the new location. This action does not take effect instantaneously and comes at a cost [29]. Additionally, the outcome of the migration action cannot be gauged until the VM is up and running the new DC. During migration, the targeted application incurs increased resource usage at the sending and receiving DCs and over the intermediate network links. The added cost and duration of a migration needs to be taken into account when re-evaluating the placement of an application. The incurred VM migration overhead is explored in Paper II. The performance conflict between the running application and the VM replication process is discussed as an ongoing work in Chapter 4.

Chapter 3

Summary and Contributions

As previously stated, the aim of this thesis is to explore the resource management challenges of the MCN. The four papers summarised below construct a chronological narrative of the work we have done so far and our on-going research. The four papers consecutively build on the progress from the previous paper and opens new research questions.

3.1 Research contributions

My primary field and contribution consistent throughout the below works is in performance modelling and the development of subsequent simulation frameworks.

3.1.1 Paper I: Workload displacement and mobility in an omnipresent cloud topology

The first paper in this thesis explores how user and demand mobility affects the resource utilisation of an MCN. The paper attempts to shed light on the challenges introduced by user and demand mobility, and reveals that intelligent resource management methods need to be developed. Furthermore, the paper introduces a coarse-grained performance model that captures user mobility, service traffic, compute resources and the consumption of those resources. The model makes the assumption that compute resources are located in the RBSs, and disregards radio channel properties. The work employs a simulator to reveal the incurred resource usage and performance degradation introduced by user mobility. The primary simulation scenario encompasses a train travelling along a one-dimensional path past a series of equidistantly spaced radio base-stations. Sessions to the service hosted in the RBSs are spawned by a body

of independent users, uniformly distributed throughout the train. Sessions and user states are migrated from RBS to RBS as users are handed over from cell to cell. The results reveal that user mobility incurs a significant resource overhead. In this work I was the primary researcher and I contributed with the model, simulation framework, scenario and analysis.

3.1.2 Paper II: Telco Clouds: Modelling and Simulation

Extending the one-dimensional model in Paper I, Paper II formalises a two-dimensional topological model with a finer grained compute resource and execution model. Furthermore, the extended model also captures the dynamics of a VM's life-cycle by modelling a VM's states. The primary contribution of this paper is the formalisation of a coarse-grained performance MCN model, a review of the currently available simulation frameworks capable of capturing the properties and dynamics of an MCN. The survey results in a proposed bespoke simulation framework that employs the models specified in the paper. The paper proceeds with evaluating the proposed model and simulation framework by studying the effects of two-dimensional user and demand mobility on DC resource utilisation. The simulation scenario consists of a uniform-grid topology where users are mobilised according to a Brownian motion model, using a variety of mobility/vehicular modes. The scale and service catchment of each DC is symmetrically varied to reveal the effects catchment has on VM migration overhead. The simulation reveals that significant gains can be made when intelligently managing VM life-cycles and migration. In this work I contributed with the network and infrastructure models, development of a simulation framework, showcase scenarios and output analysis.

3.1.3 Paper III: Resource Management Challenges for the Infinite Cloud

Built on what was learnt in Papers I and II, Paper III proposes a number of MCN research directions. Papers I and II reveal how user and demand mobility presents a non-trivial placement challenge to the operator of an MCN infrastructure. Paper III progresses the discussion by elaborating on various approaches to archiving a set of MCN management objectives, by addressing the challenges presented by a spatially dynamic demand, heterogeneous resources, and a highly distributed infrastructure. The paper begins with outlining a set of possible application behavioural and architectural traits, and progresses with outlining the challenges of evaluating the placement of each application component. Based on the accumulated experience of large scale distributed systems, the paper raises the challenge of scalability and the need for feedback in the decision process to achieve the desired management objectives.

Furthermore, the paper concludes by contrasting the challenges of a centralised versus a distributed collaborative holistic management implementation. In this paper I contributed with the formulation of the research challenges and the relevant surveys.

3.1.4 Paper IV: Dynamic Application Placement in the Mobile Cloud Network

Paper IV builds on the modelling and simulation results of Papers I and II, and attempts to address some of the challenges presented in Paper III. The paper contributes with a set of cost-based heuristics and a cost-function for optimal application placement. The proposed heuristics take into consideration resource utilisation and the cost of execution. Furthermore, the paper contributes with a set of mathematical models reflecting the resource usage of multi-tier applications and the composition of their spatial demand, the infrastructure resource usage they incur, and the performance constraints and bounds imposed by the applications and the MCN infrastructure. To enable a more general exploration of the application placement problem, the model in Paper IV departs from the models in Papers I and II by employing a flow based paradigm rather than a session and packet based simulation paradigm. The paper advances by employing the proposed model in a coarse-grained simulator to explore the performance of a set of centralised placement principals, based on the proposed heuristics. The simulation results reveal that the proposed local search algorithm performs near optimal, with a lower time complexity. The paper then proceeds to discuss the time complexities of the proposed algorithms and how they bound scalability. In this work I formulated the research question, developed the infrastructure and application models, formulated the optimisation problems and the resulting search algorithms, implemented the search algorithms and the simulator, and analysed the results.

3.2 General Conclusions and Future Work

The above papers make an attempt at addressing some of the fundamental resource management challenges found in the MCN. They draw the conclusion that the multi-dimensional objectives and dynamics of the MCN are non-trivial to accommodate at scale. In our current work we are exploring scalable application placement algorithms. We attempt to do so by first de-constructing and alleviating the time-complexity constrains for each degree of placement freedom in the centralised case. To capture the true dynamics of the system, it is essential to consider the cost of migration. Nevertheless, when attempting to do so will require that you are able to evaluate the aggregate cost of the system with an certain degree foresight. Introducing prediction is a clear step forward to advance this body of work.

Chapter 4

Non-intrusive replication for fault tolerance

Although public cloud infrastructures possess inherent redundancy and thus a certain degree of fault tolerance, cloud providers do not offer any fault tolerance guarantees. Furthermore, public DCs are vulnerable to service outages as a result of for example natural disasters and cluster and/or single Physical Machine (PM) failures. To guard against such outages, application owners opt to establish and maintain geographically distributed replicas. The replicas are dormant until the primary node fails. The replicas are continuously updated to mirror the operational main VM. The replication traffic is additive to the service traffic and share the same network resource. Therefore, there is no guarantee that the replication traffic will not interfere with the service traffic when the aggregate traffic flow exceeds the capacity of the network resource. The work presented below is an on-going work with Jonas Dürango, Amardeep Metha, Martina Maggio, Luis Tomas, and Maria Kihl.

4.1 System

To be able to resume the service at the secondary DC, both storage volumes and VM images are replicated, see Figure 4.1. They are typically transferred over then intermediate public Wide Area Network (WAN). When using a synchronous replication scheme, the latency and jitter introduced by the public WAN severely inhibits the throughput and thus so also the replication consistency [30]. Therefore, asynchronous schemes with loose consistency are preferred when replicating between geographically separated DCs.

Moreover, the asynchronous replication traffic is buffered either at the host PM

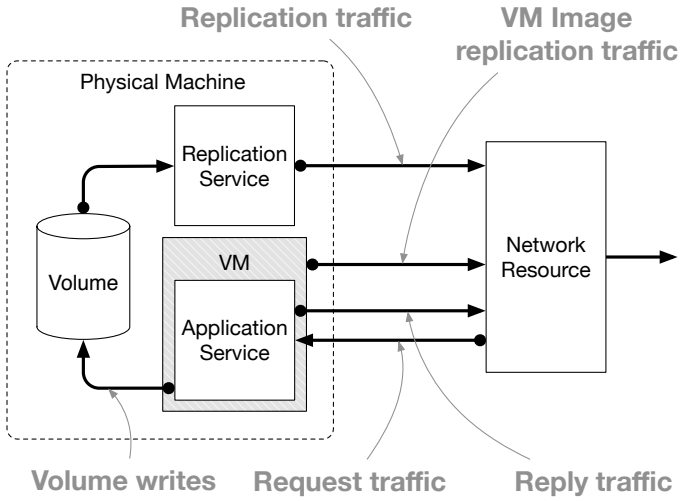


Figure 4.1: Replication system

or at an intermediate network node, such as an OpenStack Neutron. The replication traffic can for example be a consequence of users updating a local database, uploading files, or a permutation of the internal state. If the replication traffic does not achieve enough throughput its buffer will begin to populate. An overflown replication buffer halts the service process until it begins to deplete, incurring a severe drop in service response time. With a loose consistency model, the sender is not aware of the degree of consistent at the replica. The only indicator of the degree of consistency on the primary node is thus the population of the replication buffer.

The VM hosting the service and the resulting service traffic are agnostic to the replication process. The traffic it produces and consumes therefore competes with the replication traffic at the shared network resource, in a best effort manner.

4.2 Challenge

We make the assumption that the service VM is provisioned so that it is stable under a steady state workload. Without any active control the flows will compete for the shared bottleneck resource with no performance guarantees for the service.

Many of the prevailing asynchronous replica schemes employ mechanisms that operate in the Linux kernel on the host PM, such as DR.BD [31]. The schemes themselves are separate from the network stack and can therefore not practice any form of adaptive rate control on either of the competeing flows. However, the replication

scheme can restrict the rate at which it deposits data into the network stack, in a fixed throughput manner. The current state of the art is a fixed throughput cap on the replication traffic, conventionally set to 30% [31] irrespective of the service traffic rate, to which the replication scheme is agnostic. Additionally, doing so neither guarantee that the replication buffer will not overflow, nor that the available throughput is optimally utilised.

Furthermore, a Linux QDics [32] can introduce holistic traffic shaping across multiple flows by introducing a buffer for each flow and enforces a specified static proportional cap on each buffer in a, Weighted Fair Share (WFS) manner. Nevertheless, a WFS will neither ensure that the replication buffer will not overflow, nor that each traffic type meets it performance goals unless the system administrator has a priori knowledge of the onset workload.

A feedback controller that dynamically adjusts the throughput ratios would enable the system to take full advantage of the available network capacity. It would for example be able to temporarily violate the service's Quality of Service (QoS) to ensure that the replication process will not overflow its buffer and suspend the service, while taking into account the replication rate and buffer length. In this work we propose an Model Predictive Control (MPC) to dynamically manage the system's shared network resource and accommodate the individual flows' constraints.

4.3 Model and system dynamics

To study the problem we model the system as a set of time-variant traffic flows each with a corresponding buffer. Furthermore, the time-varying amount of buffered packets for service traffic is denoted $x_s(k)$, replication traffic $x_r(k)$, and VM image traffic $x_{VM}(k)$, respectively, where k denotes time steps with some sampling resolution. Let $\lambda_s(k)$, $\lambda_r(k)$ and $\lambda_{VM}(k)$ denote the incoming traffic rates for each traffic type, and $S(k)$, $R(k)$, and $VM(k)$ denote the volume of served traffic for each traffic type. We formulate the following linear buffer dynamics:

$$x_s(k+1) = \max(x_s(k) + \lambda_s(k) - S(k), 0) \tag{4.1}$$

$$x_r(k+1) = \max(x_r(k) + \lambda_r(k) - R(k), 0) \tag{4.2}$$

$$x_{VM}(k+1) = \max(x_{VM}(k) + \lambda_{VM}(k) - VM(k), 0) \tag{4.3}$$

Furthermore, C denotes the capacity of the shared network resource used for transmitting data. C can be interpreted as the available output bandwidth. The relationships between the traffic flows and the respective buffers should be interpreted as in Figure 4.2.

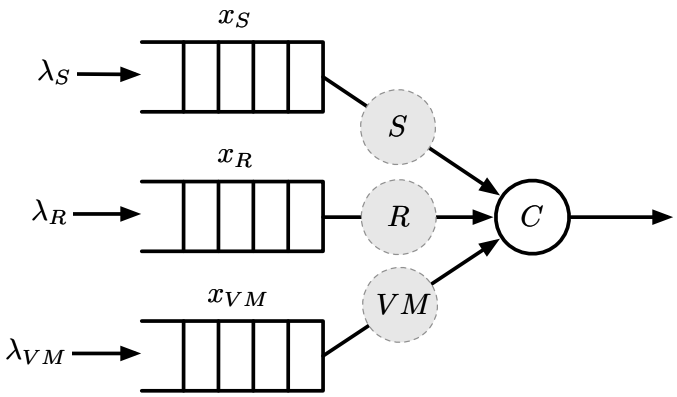


Figure 4.2: Traffic flows and buffers

4.4 Approach

We model the traffic streams as flows and employ an MPC to dynamically adjust the throughput ratios. The explicit purpose of the controller is to maintain a trade-off between high QoS, while also allocating capacity to transmit replication and VM image data. We do so by observing the length of the respective buffers as a heuristic for the latency of the service traffic and the consistency of the replication. It is natural to formulate the problem as minimising a cost function, which we formulate as:

$$J = \sum_{k=0}^{N} x_s(k)^2 + \alpha x_r(k)^2 + \beta x_{VM}(k)^2(k) \tag{4.4}$$

where α and β determine the relative weight for the different traffic streams. The cost function penalises large buffer populations, skewed by the relative weight of each traffic type. We formulate the following optimisation problem:

$$\min \sum_{k=0}^{N} x_s(k)^2 + \alpha x_r(k)^2 + \beta x_{VM}(k)^2 \tag{4.5}$$

$$\text{subject to } (4.1) - (4.3) \tag{4.6}$$

$$0 \leq S(k) \leq x_s(k) + \lambda_s(k) \tag{4.7}$$

$$0 \leq R(k) \leq x_r(k) + \lambda_r(k) \tag{4.8}$$

$$0 \leq VM(k) \leq x_{VM}(k) + \lambda_{VM}(k) \tag{4.9}$$

$$R + W + VM \leq C \tag{4.10}$$

If all packets are of equal size, (4.10) reduces to $R+W+VM \leq C$ and the optimization problem becomes convex.

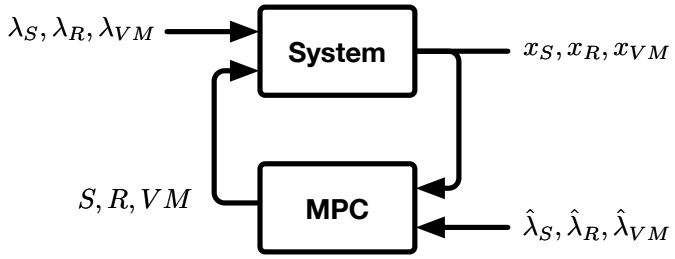


Figure 4.3: MPC Controller

4.5 Preliminary evaluation

To gauge the effectiveness of the MPC, the above optimisation problem has been implemented in a coarse discrete-time simulator using MATLAB with CVX as the solver. To contrast the proposed MPC with existing solutions we also constructed a scenario with static bandwidth allocation and one that prioritises the service traffic type. We subjected the three schemes to a sinusoidal workload:

$$\lambda_s = 3 \cdot (7 + 5 \cdot \sin(0.1 \cdot k)) \tag{4.11}$$

$$\lambda_r = 2 \cdot (7 + 3 \cdot \sin(0.2 \cdot k + \pi/4)) \tag{4.12}$$

$$\lambda_{VM} = \begin{cases} 50 & k = 20 \\ 50 & k = 60 \end{cases} \tag{4.13}$$

where the aggregate traffic occasionally exceeds C, see Figure 4.4.

In the experiment below we used the MPC specified in above, with $\alpha = 0.1, \beta = 0.00002, \text{horizon} = 2$. Figure 4.5 reveal that the MPC approach sacrifices service throughput in favour of an unsaturated replication buffer. Given that some service traffic buffering is not guaranteed to violate the service QoS, this behaviour can be tolerated when holistically managing the systems resources and constraints. Additionally, the MPC solution achieves a near-optimal network resource utilisation.

4.6 Future work

As a first step to continue this work, for greater granularity, we will implement and run the experiments in an event-driven simulator. To further study the properties of

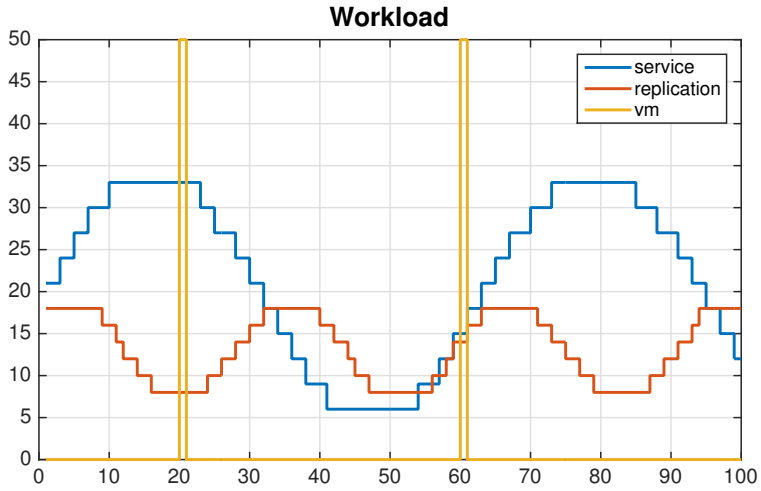


Figure 4.4: Workload

the MPC approach we will need an expanded set of workloads to cover a wider range of operational scenarios and introduce a set of performance metrics to quantitatively contrast the relative performance of the MPC approach with exiting solutions. Finally, we intend to implement and verify the MPC approach in a test-bed running DR.DB.

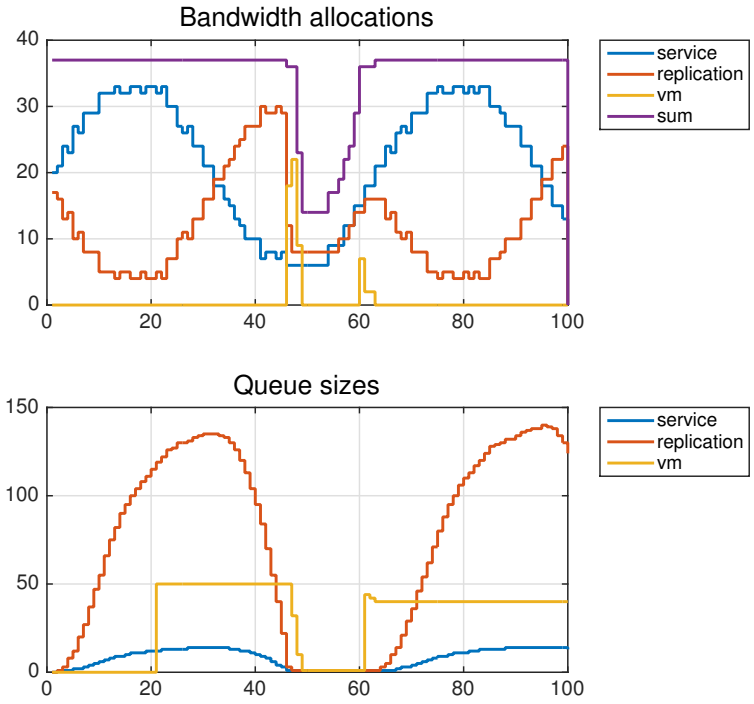


Figure 4.5: MPC

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov 2008, pp. 1–10.

[2] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.

[3] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@ home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] A. Amazon, "Amazon web services," *Available in: http://aws. amazon. com/es/ec2/(November 2012)*, 2010.

[6] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, "Microsoft azure and cloud computing," in *Microsoft Azure*. Springer, 2015, pp. 3–26.

[7] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *Personal Communications, IEEE*, vol. 8, no. 4, pp. 10–17, 2001.

[8] P. Bosch, A. Duminuco, F. Pianese, and T. L. Wood, "Telco clouds and virtual telco: Consolidation, convergence, and beyond," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 982–988.

[9] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[10] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, ser. MMSys '10. New York, NY, USA: ACM, 2010, pp. 35–46. [Online]. Available: http://doi.acm.org/10.1145/1730836.1730842

[11] P. Brebner and A. Liu, "Modeling cloud cost and performance," in *Proceedings of Cloud Computing and Virtualization Conference (CCV 2010), Singapore*. Citeseer, 2010.

[12] H. Basilier, M. Darula, and J. Wilke, "Virtualizing network services–the telecom cloud," *Ericsson Review*, 2014.

[13] H. Guan, T. Kolding, and P. Merz, "Discovery of cloud-ran," in *Cloud-RAN Workshop*, vol. 2010, 2010.

[14] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud ran for mobile networks—a technology overview," *Communications Surveys & Tutorials, IEEE*, vol. 17, no. 1, pp. 405–426, 2014.

[15] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[16] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science, C.-H. Hsu, L. Yang, J. Park, and S.-S. Yeo, Eds. Springer Berlin Heidelberg, 2010, vol. 6081, pp. 13–31. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13119-6_2

[17] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4:1–4:11, July 2009.

[18] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1587–1596 vol.3.

[19] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1455–1468, Sept 2011.

[20] P. Svärd, W. Li, E. Wadbro, J. Tordsson, and E. Elmroth, "Continuous datacenter consolidation," ser. Report / UMINF, no. 2014:08. Umeå universitet, 2014, p. 12.

[21] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[22] F. P. Tso, G. Hamilton, K. Oikonomou, and D. Pezaros, "Implementing scalable, network-aware virtual machine migration for cloud data centers," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, June 2013, pp. 557–564.

[23] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services." in *USENIX Symposium on Networked Systems Design and Implementation*, 2010, pp. 17–32.

[24] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 74–85, 2011.

[25] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384–392, 2002.

[26] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. Ieee, 2009, pp. 44–51.

[27] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1071690.1064252

[28] H. Goudarzi and M. Pedram, "Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 324–331.

[29] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Cloud Computing*. Springer, 2009, pp. 254–265.

[30] S. Rajagopalan, B. Cully, R. O'Connor, and A. Warfield, "Secondsite: disaster tolerance as a service," in *ACM SIGPLAN Notices*, vol. 47. ACM, 2012, pp. 97–108.

[31] F. Haas, P. Reisner, and L. Ellenberg, "The drbd user's guide," *LINBIT HA Solutions GmbH*, 2011.

[32] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, "Linux advanced routing & traffic control," in *Ottawa Linux Symposium*, 2002, p. 213.

Part II

Included Papers

Paper I

Workload displacement and mobility in an omnipresent cloud topology

Latency and throughput demands on cloud hosted services are growing more complex as cloud services are at an increasing rate being consumed on mobile devices. On mobile devices, cloud services are accessed through a WAN and a mobile access network, through which latency is added and throughput restricted, resulting in an inconsistent user experience. The proposed omnipresent cloud topology paradigm attempts to remedy this latency decay by placing generic cloud data centres, of arbitrary size, in closer geographic proximity to the end user, thus reducing the geographic discrepancy that contribute to congestion and latency. A key performance challenge in the omnipresent cloud paradigm is the incurred cost of service migration as a result of user mobility. In this paper we examine fundamental resource costs and dynamics of user mobility in an omnipresent cloud topology. Furthermore, this paper also propose and evaluates a simulation model capturing the fundamental dynamics of an omnipresent cloud architecture in an extreme operating scenario. Our simulations reveals that mobility significantly affects the proportion of sessions that are migrated between consecutive nodes and that migration can consume up to 20% of the systems resources.

William Tärneberg, Maria Kihl,
"Workload displacement and mobility in an omnipresent cloud topology"
IEEE SoftCOM 2014, Split, Croatia, Sept 2014.

1 Introduction

With the arrival of more seamless and accessible cloud services, they are growing more popular with mobile users. Cloud services have matured to the point where they range from on-line storage, to biometric monitoring, to grid system management, to crowd collaboration, to big data collection, to small web services. Latency sensitive services such as industrial process control, game rendering, and financial trading have so far, given existing infrastructure, in many cases not been candidates for cloud migration [1]. Mobile oriented cloud services have evolved to a point where they behave like traditional device-centric services, such as storage and mobile applications. As a result, the intermediate delay between the device and the data centre is a crucial factor in the perceived performance of the services.

Cloud service performance and so also their potential prevalence is constrained by the best-effort network it is delivered through. WAN latency and throughput bottlenecks have an observably clear correlation with the geographic discrepancy between the cloud hosting infrastructure and the end-user, being it wireless or wired [2]. Additionally, [1] shows that VM interference and traffic congestion when hosted in large, resource-ubiquitous, data centres can have a detrimental effect on a service latency.

Under the presumption that the geographic disparity between the user and the host correlates with latency and throughput, various research efforts are being directed at accommodating cloud services in the emerging, all-IP (Internet Protocol), next generation networks [3, 4]. Essentially, the proposed paradigm shift relocates or co-locates cloud data centres progressively towards the capillaries and edges of the mobile access networks. More specifically, a smaller data centre or server adjacent to a radio base station can proposedly host Virtual Machines (VM) to which one or multiple users can subscribe. To minimize the distance between the subscribers and the data centre, the hosting VM will appropriately be migrated and/or duplicated geographically with its subscribers. Alternatively, services can be hosted in aggregated data centres, serving subscribers from multiple radio base stations, depending on the topology of the mobile access network. In this paper we refer to this topology paradigm as the omnipresent cloud.

User mobility is a key differentiator between traditional data-centre centric clouds and the omnipresent cloud. In the omnipresent cloud, a user's location, within a few meters or kilometres, determines in which data centre a service is executed. The rate of its user's movement determines to what extent and to where a service needs to be migrated in order to achieve a desirable latency level.

The scope of much of the existing research and literature related to omnipresent clouds is in the context of network virtualization [5, 6], Software Defined Networks (SDN), and Cloud Radio Access Network (C-RAN) and has focused much of its attention on IaaS solutions. Virtualization and SDN will strongly characterize the development of the next generation of mobile networks by making the networks more dis-

tributed and its nodes more autonomous by granting them more centralized compute and software capabilities. These infrastructure resources, now distributed throughout the network, can proposedly host other services than those procured with maintaining the network. There are numerous papers [7, 8, 4] dedicated to exploring plausible economic and IT models of such schemes, producing protocols and IT/IaaS solutions, such as TSaaS [8]. The primary intent of these proposed solutions is to reduce operational expenditure, increase service deployment flexibility as a means to increase the speed of which services can be introduced to the network.

However, the relationship between service performance versus geographic location has received much less research attention than that directed at the added technical deployment and revenue flexibility the proposed IT solutions might contribute [9, 10]. There is thus comparatively little research bridging state of the art cloud hosting research and a clouds ability to operate in a mobile network with mobile users. What is specifically lacking is how the mobile user generated workload will vary and be displaced between the omnipresent cloud data centres as a consequence of user mobility and a study of the associated resource cost. Furthermore, [2] investigates data-center latency in geo-distributed networks in the context of the operational cost of transmitting and operating the intermediate network at a desirable performance level. The authors of [11] studied the effect of migrating user instances geographically to existing geo-distributed data center, in response to a users location on a global, inter/intra-continental scale. However, in the omnipresent cloud, user movement is potentially significantly more rapid across the associated data centres.

In this paper we explore the fundamental dynamics of workload displacement as a result of user mobility between independent data centres adjacent to and associated with a radio base station. We proceed with examining the proportion of workload being displaced to adjacent data centres, and the proportion of resources the act of migration consumes in a data centre in relation to the work it completes. We also propose a simple simulation model that includes the basic building blocks of the omnipresent cloud, in conjunction with a mobility model aimed at provoking and exploring basic system workload displacement vulnerabilities and the dynamic effects on service performance as a result of mobility.

Our results show that user mobility in an omnipresent cloud topology prompts a cumulative spatial displacement of workload in successive server nodes. Additionally, when the server nodes are over-provisioned, our simulation reveals that the server nodes, at an increasing rate, spend more time migrating VMs than executing them. As a result, a stable system-wide waiting time is only attainable with a system load of less than 80%. The simulations also reveal that despite a stable system, the waiting time still increases in the spatial domain as a result of user mobility. The paper also investigates a user's utility in subscribing to an omnipresent cloud node.

Section 2 outlines the fundamental principals of the proposed omnipresent cloud topology, while Section 3 details which aspects and abstractions of the omnipresent

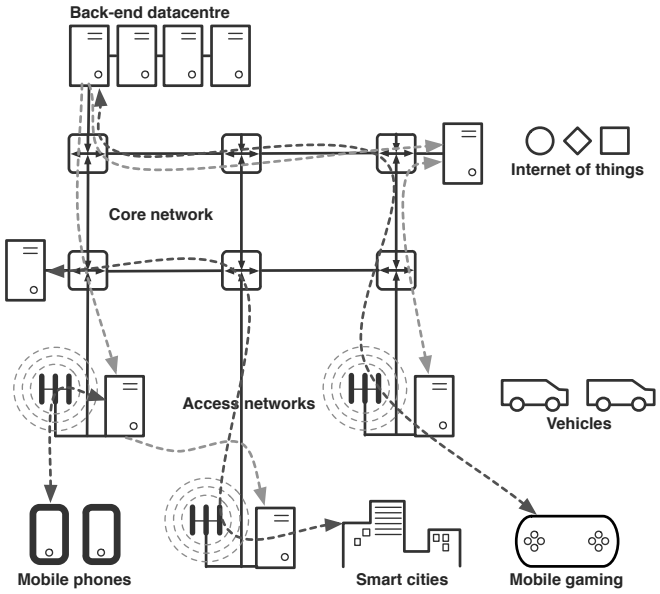


Figure 1: Omnipresent cloud

cloud topology that are included in our experiments. Furthermore, the resulting simulation model and its constituent parts are specified in Section 4 followed by Section 5, which accounts for the specifics of the simulation experiments. Lastly, Sections 6 and 7 present the results and consultations drawn from the experiment.

2 Omnipresent cloud

The omnipresent cloud paradigm presents a novel, Telecom-centric, way to remedy the latency the WAN between the cloud data centre and its users introduce. Cloud services can range from IaaS, PaaS, SaaS, SECaaS, to simple web like services. Furthermore, due to the great variety of services that collectively coexists in the mobile network, each instance of a service node plausibly hosts several heterogeneous services, each contained within a Virtual Machine (VM) or Container. In the extreme case, for example, when arbitrary code is offloaded from a mobile phone [12], each VM might serve just one user. As illustrated by Figure 1 an omnipresent cloud topology's hardware resources are positioned with close proximity to the user throughout the mobile network.

To maintain proximity to the user as it moves around the network, the service

migrates geographically with the user to the closest node. Proposedly, where omnipresent cloud infrastructure is available, a service instance can be migrated from a distant data centre where it traditionally resides to an omnipresent cloud node in the mobile network. As mobile users move through the network, and when it is deemed optimal to migrate a service given a geographic discrepancy, the concerned VM is migrated to where latency and congestion is minimized. However, doing so will incur an additional load both on the receiving and sending nodes, and the intermediate WAN. Moreover, migration and overhead is minimized if the amount of work completed in each node is maximized during a user's residency, and when inter-data centre transmission is minimized.

3 Targeted scenario

In this paper, we propose a simulation model and evaluate basic complexities introduced by user mobility in an omnipresent cloud topology. Our investigations are performed by simulation using the model shown in Figure 2. To explore the extreme scenario, in this paper, to strictly minimize the proximity to the user, each abstract radio base station will host a cloud server entity.

In order to be able to observe consecutive workload displacement, users are displaced according to a train model at constant speed along a linear path though a one-dimensional space. Furthermore, throughout the one-dimensional space, radio base stations are equidistantly positioned.

In our proposed model, user movement and network resources are homogeneous. As a result, we will be able to observe the proportional displacement of workload between comparable service nodes, as users move between nodes. Additionally, this will also reveal the subsequent proportional degradation of perceived service quality, experienced by the user over the whole network. We will also be able to discern the rate of which a service needs to be migrated, which can be seen as an abstract measure of the scale of a resulting VM or Container migration. The simulation will reveal how mobility affects the proportion of sessions that will be migrated between consecutive nodes, consecutive degradation of waiting time, and the potential resulting VM migration burden imposed on the system.

4 Simulation model

Our discrete time simulation model contains multiple independent users N_u, each with a unique location determined by a train mobility model. The users location within a network determines which singular radio base station it is associated with.

The modelled network contains multiple, equidistantly separated radio base stations. Each radio base stations or cell has a fixed coverage radius, r_{cell}. The network re-evaluates user and radio node association at a certain rate throughout the simulation. All user generated requests are sent to its current associated radio base station. The radio node forwards subsequently all incoming requests to a singular server node which processes the incoming requests at a certain service rate $T_{service}$.

4.1 Service model

The adopted service model is based on the open-loop, one tier, long tailed, HTTP request mode detailed in [13]. The modelled traffic is consistent with web surfing on mobile devices, where users access mobile-adapted web pages with very little inline dynamic content, revisited at a high frequency. Additionally, the duration of the resulting sessions is proportional to the radius of the networks radio cells. Each session spawns a number of requests proportional to the File size (S_f) and the Request size (S_r) in KB, both Pareto distributed. Each request is separated by a Inter-request Weibull-distributed delay (D_r). Moreover, each session is separated in time by an Pareto distributed inter-session delay (D_s).

4.2 Network model and topology

All radio access nodes are contained within a network entity, each radio base station is bounded by a cell coverage radius, r_{cell}. Given that a user is within the aggregated cell coverage of the network, that user will always be associated with the radio access node closest to it. The network periodically evaluates each user's proximity to all the radio access nodes in the network. If a user moves closer to another radio access node, at that threshold, a handover will occur and the radio access node association will be updated, see Figure 2.

4.3 Mobility model

Our simulation model uses a train mobility model, which operates in one dimension, clusters N_u users, and and displaced its objects at a constant velocity, V_{train}. A train model presents a extreme mobility condition where the total user population and thus traffic is displaced in concentrated groups from node to node, progressively and permanently abandoning radio base stations in rapid succession.

4.4 Server node model

Each server node is modelled as a single server queue that processes requests from its deferred queue with an exponentially distributed service time $T_{service}$. Furthermore,

when a user is handed over from one radio node to another, all deferred requests from that user in the active server node queue are instantly migrated to the newly associated server node. More precisely, this occurs when the current process is completed, and incurs no additional load to the network or the server. The migrated requests are placed at the end of the receiving server node's queue. Any ongoing processing is completed before the migration procedure begins.

The mechanisms the govern the provisioning of network resources and cloud resources are in this model, independent. The association and connection between a radio access node and a server node is arbitrary and is not specific to any particular mobile system generation topology.

5 Experiments

The adopted simulation model was implemented as a discrete event Java simulator using simjava [14] as the event engine. In order to be able to evaluate geographic load displacement and the subsequent service performance degradation in relation to server load scenarios, using the model above, server load levels at 50% to 150% were deployed in the simulation model. In this paper, server load is defined as the inverse percentage of the request service time $T_{service}$. Moreover, the request service time is defined as the quotient of the total arrival rate at full user residency, see Equation 1, where λ_i is the arrival rate for the ith user. For example, a 50 % load is when $T_{service}$ is twice as high as the aggregate inverse arrival rate.

$$T_{service} = \frac{1}{\sum \lambda_i} \tag{1}$$

In order to ensure that the system is subject to multiple migrated sessions, the mean service session duration is set proportional to the radius of a cell, r_{cell}. As a result, all requests equal to and below the mean session length will on average be completed in one server node, while those above, will on average, be subject to migration. Given the previously mentioned radio node displacement and user spatial density, each user will be associated with and reside within the domain of each radio access node for 40 seconds.

The simulation runs for T_{sim} minutes, through which the train of passengers pass through 7 radio base station domains. Given the service model described in Section 4.1, the simulation reaches its steady state after 3.6 simulation minutes, at which point the first user gets in range of the first radio base station. Consequently, the total steady state simulation time amounts to 5,2 simulation minutes. The steady state simulation time is sufficient to allow each user to spawn several open-loop sessions and thus to reveal the fundamental dynamics of the system. Designedly, the first radio node will not be subject to migrated requests.

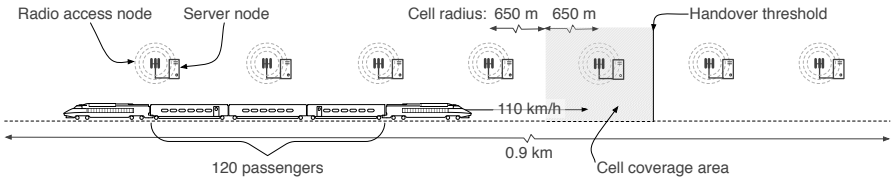


Figure 2: One dimensional simulation scenario

Table 1: Simulation parameter values

Parameter	**Value**
r_{cell}	650 m
N_u	120
V_{train}	110 km/h
$T_{service}$	50-150% of 0.0039 seconds
T_{sim}	8,8 minutes (7 nodes)

The simulation scenario will include several server load levels. Feasibly, homogeneous server nodes subject to a load greater than 100% will result in an unstable system with a transient workload growth. Given a certain user velocity, an unstable system will result in varying service response times with displacement. Note that, as we modelled the system without signalling latency, the waiting and service times can be regarded as the server response time.

Furthermore, service model parameters are sampled from the distributions in Table 2 in accordance with [13]. Similarly, Table 1 details the global simulation scenario parameters.

Each server node was sampled for; queue length, waiting time, and processed and migrated request sizes per session. These parameters allowed us to reveal how mobility affects the proportion of sessions that will be migrated between consecutively nodes, consecutive degradation of waiting time, and the potential resulting VM migration burden placed on the system. The resulting data is comprised of the mean of 10 independent replications.

6 Results and discussions

In this section we present the results from the simulations and their implications. Figure 3 shows how workload is spatially displaced when server nodes are subject to a load greater then 100%. As users move out-off and in range of subsequent server

Table 2: Service model components

Component	Distribution	Parameters
S_f	Pareto	K=133000 α =1.1
S_r	Pareto	K=1000
D_r	Weibull	α =1.46 β =0.382
D_s	Pareto	K=1 α=1.5

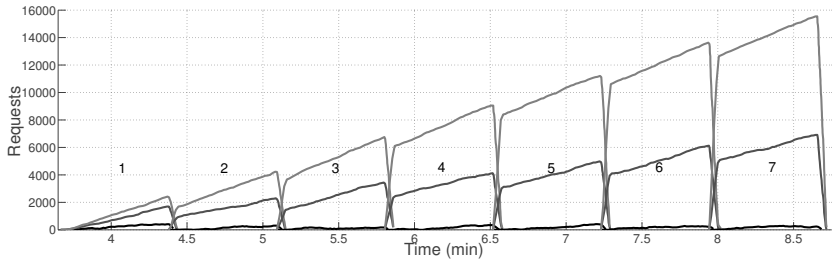


Figure 3: Queue length displacement at 100%, 110%, and 120% load, respectively. Each node is marked with its corresponding consecutive number.

nodes, any incomplete requests will be migrated to the subsequent server node. The average deferred queue length exhibits growth according to $c \cdot n_i^l$, where n_i is the ith node and l the load quotient, e.g. 120% = 1.2. Additionally, given that the sessions are longer than the duration a user spends in radio base station and data centre pair, the subsequent nodes will need to, on average, be able to absorb the additional migrated load.

Figure 3 reveals the load point where the system becomes unstable. Any server load greater than 100% of the homogeneous server nodes result in an unstable system with a progressive degradation of waiting time. As a consequence of user mobility, a cumulative amount of workload is migrated to the subsequent nodes to the point where the system is unable to recover.

Furthermore, note that Figure 3 shows how the deferred queue length at 100% load grows during maximum user residency to the point where sessions are not completed and are thus migrated to the subsequent node. Nevertheless, both the sending and receiving nodes are able to recover during the transitions between nodes, and thus maintain stability.

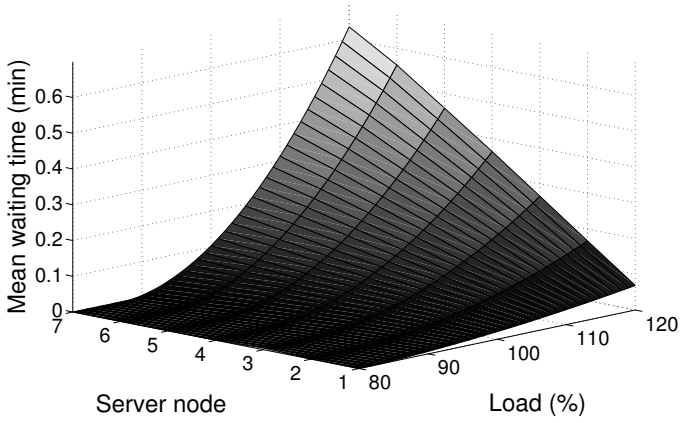


Figure 4: Waiting time degradation

6.1 Waiting time degradation

Degradation of waiting time is another consequence of the above-mentioned progressive workload build-up. This occurs when the server nodes are subject to loads greater than 80%, which is shown in Figure 4. As can be seen, the mean waiting times during max residency increase linearly for each consecutive server node. A user will thus experience a linear degradation of the mean response time in space. In addition, the mean waiting times for each server node as a function of the load level grows quadratically with increased load.

As illustrated by Figure 3, at the maximum stable load (100%), beyond which, the queue length diverges, the system is able to maintain a consistent deferred queue length and session residency, but because of migration and the resulting session migration effort, waiting time degrades 5 fold across the span of the network. Only at a load of less than 80% is the system able to recover the incurred migration effect and thus maintain a consistent waiting time. This implies that in order to maintain system stability, the individual server nodes can never be provisioned to utilize 100% of its resources.

6.2 Session and VM migration

We showed above that request migration incurs a degraded response time. Furthermore, each of those requests constitute a subset of a session. As detailed earlier, in this paper, each session is regarded as a VM instance in a generic cloud server. As such, observing the residence and migration of sessions reveals how often VM migra-

tion occurs and the potential load a VM migration can incur.

Our investigations show that at 100% load, 90% of the VM are completed in one node and are not subject to migration. On the other hand, at 120% load, on average, a VM in the last of the 7 nodes only completes 10% of its request, the corresponding value for the first node is 20%. Moreover, at a 120% server load, on average 65% of the incoming requests receive 0% of that node's compute cycles. In other words, some VMs do not receive any resources to complete any of its requests despite the system spending resources migrating these VMs to the next node. At this point the paradigm is contributing far more latency than it is eliminating.

6.3 VM migration time

In terms of the VM migration time, in order to maintain a consistent waiting time and allow a migration to recover, VM migration needs to be performed within the time period of the mean waiting time. The simulation discloses that waiting time recovery is only feasible at less than 80% load, and is only fully able to do so when the system is subject to a load less than 50%.

6.4 Request migration

In contrast to sessions or VMs, the rate at which requests are processed versus user node residency is a metric of utility. Figure 5 displays the proportion of processed requests that were generated in the domain of that server node. The figure reveals that the total received requests decays exponentially with each subsequent cell. At 120% load, the first node processes 90% of the requests generated in while associated with that node. The rate diminishes to 8% in the final node. Feasibly, the utility of subscribing to that node is negligible. Moreover at 100% workload, the amount of requests being processed that were generated while subscribing to that node, decays faster than the amount of migrations, which quickly converges. This behaviour is a contributing factor to why the waiting time is decaying in an otherwise stable system, as discussed above.

Consequently, the amount of time spent processing migrated requests by each individual node grows exponentially, converging to where no intra-node generated requests are processed. At this point the migrated VMs contribute more requests than what is generated within the domain of the server node. It would arguably be more efficient to eliminate much of the migration by consolidating multiple server nodes and spend those resources on processing tasks.

Furthermore, the mean waiting time in proportion to the time spent in the domain of a node gives you one metric of how much work or effort is being contributed by that node. At the far node, at 120% load, almost the whole residency is rewarded with, on

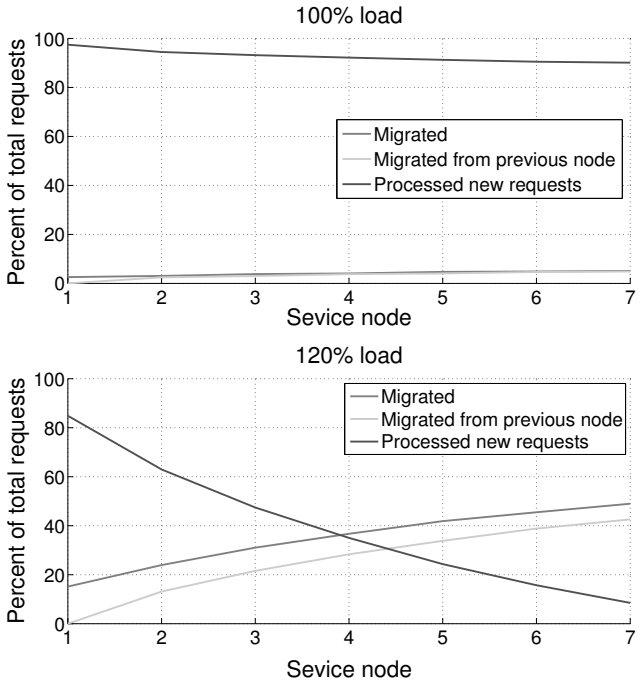


Figure 5: Migrated vs. processed packets

average, 1,11 processed requests. As such, using that node carries very little return. The effect of diminishing return of the time spent in a node is shown by Figure 5.

6.5 Session migration versus node residency time

Another relevant comparison is that of session migration versus node residency, which correspond to the general scale requirements of the resources. As one can expect, in a stable system the number of VMs will remain relatively constant over time, given a 100 % workload. It is made evident by Figure 3 that the system is able to recover from temporary overloads in one node, as any excess workload is gradually spread to the adjacent vacant nodes. This self-balancing effect is of course proportional to the distribution of users, the speed of which they are moving in and the dimensions of the radio cells.

7 Conclusions

The omnipresent cloud model and simulation reveal the challenges facing mobility in the omnipresent cloud. The simulation results made it apparent how mobility incurs severe progressive workload accumulation, and that VM migration will contribute to a large overhead, depending on the topology. The incurred VM migration load on the system consumes such a large proportion of the systems resources that it will require the system administrators to greatly over-provision the system in order to maintain consistent performance.

It was also made clear that the return of subscribing to the closest omnipresent cloud node has a diminishing utility with node order and server load. At the simulated extremes, slightly more than 1 request is processed during the time a user on average spends in a cell. Thus the cost of migrating the session far exceeded the amount of work it contributes.

Complementary, it will conceivably be relevant to determine network topological placement of the omnipresent cloud server nodes and determine the effects of services and VMs migrating to and from a distant data centre and horizontally in the network, and though other network access media, such as 802.11, as a means to load balance the system of distributed data centres.

References

[1] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 35–46.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[3] M. Gutierrez and N. Ventura, "Mobile cloud computing based on service oriented architecture: Embracing network as a service for 3rd party application service providers," in *Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services (K-2011), Proceedings of ITU*, 2011, pp. 1–7.

[4] G. Caryer, T. Rings, J. Gallop, S. Schulz, J. Grabowski, I. Stokes-Rees, and T. Kovacikova, "Grid/cloud computing interoperability, standardization and the next generation network (ngn)," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, 2009, pp. 1–6.

[5] F. Baroncelli, B. Martini, and P. Castoldi, "Network virtualization for cloud computing," *annals of telecommunications-annales des télécommunications*, vol. 65, no. 11-12, pp. 713–721, 2010.

[6] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, 2009.

[7] "The telecom cloud oppertunity," Ericsson, Whitepaper, 2012.

[8] S. Pal and T. Pal, "Tsaas; customized telecom app hosting on cloud," in *Internet Multimedia Systems Architecture and Application (IMSAA), 2011 IEEE 5th International Conference on*, 2011, pp. 1–6.

[9] V. Sarathy, P. Narayan, and R. Mikkilineni, "Next generation cloud computing architecture: Enabling real-time dynamism for shared distributed physical infrastructure," in *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, 2010, pp. 48–53.

[10] X. Zhiqun, C. Duan, H. Zhiyuan, and S. Qunying, "Emerging of telco cloud," *Communications, China*, vol. 10, no. 6, pp. 79–85, 2013.

[11] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services." in *USENIX Symposium on Networked Systems Design and Implementation*, 2010, pp. 17–32.

[12] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. S. Lee, "ucloud: Towards a new paradigm of rich mobile applications," *Procedia Computer Science*, vol. 5, no. 0, pp. 618 – 624, 2011, the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911004054

[13] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 151–160, 1998.

[14] (2014, 02) simjava. School of Infomatics, University of Edinburgh. Available online at http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/. [Online]. Available: http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/

Paper II

Telco Clouds: Modelling and Simulation

In this paper, we propose a Telco cloud meta-model that can be used to simulate different infrastructure configurations and explore their consequences on the system performance and costs. To achieve this, we analyse current telecommunication and data centre infrastructure paradigms, describe the architecture of the Telco cloud and detail the benefits of merging both infrastructures in a unified system. Next, we detail the dynamics of the Telco cloud and identify the components that are the most relevant from the perspective of modelling performance and cost. A number of well established simulation technologies exist for most of the Telco cloud components, we thus proceed with surveying existing models in an attempt to construct a suitable composite meta-model. Finally, we present a showcase scenario to demonstrate the scope of our Telco cloud simulator.

Jakub Krzywda, William Tärneberg, Per-Olov Östberg, Maria Kihl, Erik Elmroth,
"Telco Clouds: Modelling and Simulation,"
CLOSER. 2015), Lisbon, Portugal, May, 2015.

1 Introduction

Recent technological developments have enabled a union of telecommunication and cloud computing. Joint management of telecommunication infrastructure, such as RBS, and DC may help to achieve better performance of hosted applications and reduce the operation costs. In this paper we refer to this paradigm as Telco cloud computing [1].

Despite the interest in this paradigm, there are no simulation models capable of simultaneously modelling the dynamics of Mobile Devices (MD), placement and capacity of DCs, and network infrastructure. Understanding of these relations is important for Telco cloud stakeholders, e.g., Infrastructure Provider (IP) can use that knowledge to reduce infrastructure costs, while still delivering competitive performance.

We propose a meta-model of the Telco cloud which facilitates experimentation and evaluation of possible configurations, such as placement and capacity of DCs in a joint telecommunication-cloud infrastructure. The meta-model uses existing, well established simulation models, e.g., for Radio Access Networks (RANs) or DCs, for modelling of the aforementioned individual parts of the infrastructure behaviour.

The contributions of this paper are: describing the dynamics of the Telco cloud, including QoS and the associated costs of this paradigm (Section 4); surveying existing models of Telco cloud building blocks (Section 5); and establishing a meta-model that captures the described dynamics using existing and composite models (Section 6).

This paper is structured as follows. Section 2 outlines the architecture of the proposed Telco cloud. Next, the simulation motivations, challenges, and requirements that the meta-model has to fulfill are presented in Section 3. Section 4 describes the Telco cloud dynamics, followed by a survey of existing models in Section 5. Section 6 introduces the Telco cloud meta-model. Section 7 presents a showcase simulation scenario, using a prototype implementation of the introduced meta-model. In Section 8 we list a few relevant research topics that can be explored using the proposed model and conclude the paper.

2 The Telco cloud Architecture

In this section we present issues with current telecommunication and cloud infrastructures, an overview of the proposed Telco cloud topology, and how the Telco cloud paradigm can help to remedy these issues.

2.1 Current Infrastructure

Currently, telecommunication and cloud computing infrastructures are separated and managed independently. The telecommunication infrastructure is placed in close

proximity to end users and is built using specialised hardware. The cloud computing infrastructure consists of remote DCs that are significantly geographically separated from end users, consists of commodity hardware, and is connected with the telecommunication infrastructure via the Internet.

We identify several issues of the current infrastructure. Performance (especially latency) of cloud services is not predictable, which makes computation offloading difficult. All data processed in the cloud need to be sent over the Internet to DCs, which add communication latency. For the Internet of Things, with millions of sensors generating huge amounts of data, the volume of traffic can cause network congestion. Specialised telecommunication hardware is expensive and hard to upgrade.

As a consequence of the performance bottlenecks, particularly latency sensitive applications such as industrial process control and Augmented Reality context recognition applications have mostly not yet been cloudified. The low latency, jitter free, and high throughput connections required by such applications cannot be provided by the existing telecommunication and cloud infrastructures [2]. Moreover, compute and battery resources in MDs are limited and coupled. An approach to augmenting MD's capabilities is to offload the execution of applications to a cloud infrastructure. Such performance augmentation can only be seamless if the incurred communication latency is low enough and service availability is high.

Devices are at an increasing rate gaining access to the Internet [3]. Anything from small sensors to petrol pumps, flowerpots, helmets, tumblers, windows, and spark plugs is being connected to the Internet to communicate and monitor its quantified individual performance metrics. Most of the connected devices are generating correlated contextual information, incurring large amounts of WAN traffic. The traffic typically converges to a handful of DCs for analysis and processing which introduces congestion in the capacity-sparse and increasingly congested RANs, core networks, and WANs. Additionally, a portion of the transported information is only locally relevant and no or very little entropy to the service in the Remote DC.

Wireless access network virtualisation and cloudification of Telecom equipment and services proposed by [4], requires careful placement of compute capacity as not to introduce significant propagation delay. The placement of the compute nodes has to reflect the demand for Telecom- and cloud-services in the geographic area which the RAN covers. Current Telecom signalling standards and remote DCs do not interoperate well and are often not able to meet Telecom latency requirements.

2.2 Introducing the Telco Cloud

A Telco cloud is an infrastructure consisting of MDs, stationary devices (sensors), access networks, intermediate WANs (connecting the access networks to the backbone networks), backbone (Internet) networks, and DCs. We here include two main types of DCs: Remote Data Centres (large DCs located far from the access networks) and

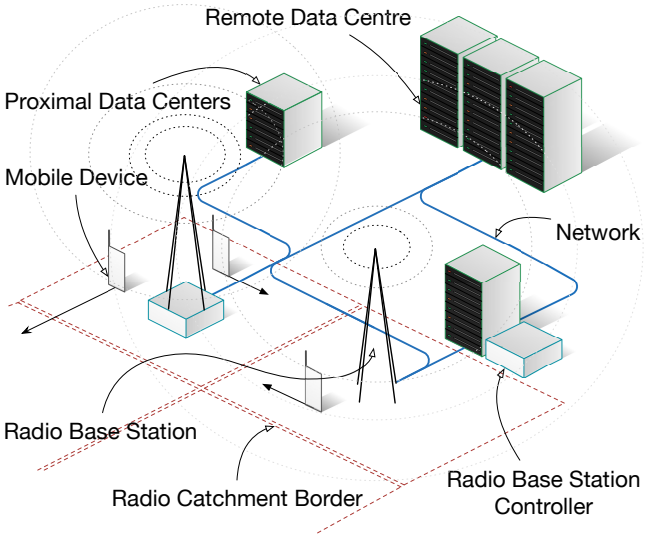


Figure 1: Overview of Telco cloud.

Proximal Data Centres (smaller DCs located close to the access networks).

The Telco cloud topology paradigm proposes a closer integration between access networks and a cloud infrastructure than the current topological model where the Telecom and cloud infrastructures are unaware of each other. When coexisting with cloud infrastructure, telecommunication functionality can be virtualised and augmented to the adjacent DCs. When virtualising RAN functions large portions of an RBS and the RAN control functions can be executed in a DC [5].

Cloud capacity will reside in geographically distributed DCs. The Telco cloud topology is composed of multiple DCs that are dispersed in a mesh-structure, ranging from complete adjacency with the Telecom infrastructure, Proximal DCs, to more traditional, Remote DCs, as depicted in Figure 1.

A group of Telco cloud DCs can be provisioned and load balanced as one resource or act as independent DCs. In the former case, a control plane will need to coordinate services and the shared resources by for example optimising locality and proximity to all entities by geographically placing services accordingly. Proximal DCs will thus have to be managed in a distributed and coordinated manner.

The cloud services and Proximal DCs are accessed through the RAN and cater for the MDs within it. The MDs are assumed to possess varying degrees of mobility, with an equivalent likelihood of passing between a particular set of macro/micro-RBS[1] over

[1] What is considered a traditional rural/urban cell, constituted by a high power RBS, mounted on a tower.

time. In an effort to be able to enforce DC management constraints and to globally avoid unnecessary migration, an MD is not strictly associated with the closest DC or the DC that its current RBS cell is associated with.

The Telco cloud infrastructure topology will need to reflect the capacity and latency objectives of the virtualised RBSs, and to give access to the Telco cloud-hosted services given the networks local capability and diversity at a sufficient service level. The prevailing 4G/LTE topology is centred around hierarchical macro- and micro-cells which very much resembles that of its predecessors. The Telco cloud topology will evolve with mobile access technology generations, shifting and/or dispersing compute capacity at various levels of mobile, Metropolitan Area Network (MAN), and WAN networks to best suit the prevailing services and throughput channels.

2.3 Benefits of the Telco Cloud

We identified four main benefits of the Telco cloud. First, it provides cloud applications with better and more predictable performance. Second, it supports computation off-loading for resource-bounded MDs. Third, the Telco cloud reduces network utilization by processing part of data closer to its producer or consumer. Fourth, it enhances cost-efficiency and flexibility of telecommunication infrastructure.

Thanks to a geographically distributed cloud infrastructure, application developers and telecommunication operators can take advantage of the significantly lower round-trip times. Moreover, we expect that the average network throughput will increase when communicating with Proximal DCs in comparison to Remote DCs. Additionally, users offloading MD applications will benefit from a low latency communication with a DC, where the code is executed.

The large amount of information generated by sensors can be filtered through the intermediate hierarchical cascade of DCs to prevent congestion in the intermediate WAN. Data that is only locally relevant can be kept and consumed locally, while redundant and highly covariant information can be more easily identified in a local context and discarded.

Through the Telco cloud traditional proprietary hardware-bound telecom services can be virtualised, migrated, and executed in Proximal DCs. Multiple RBSs can be consolidated to increase the aggregate utilisation of the infrastructure. Executing RBSs on cloud infrastructure will allow for greater use of cost-effective commodity hardware and generic software. With the availability of the Telco cloud, we expect that an RBS in future mobile infrastructure generations will only consist of a radio interface. The management of the RAN, individual radio channels, signalling, services, and signal processing, can be all virtualised and executed in a Proximal DC in the vicinity.

3 Simulation challenges

Simulation of a Telco cloud is motivated by several factors, e.g., lack of existing infrastructure and appropriate control plane standards. The desired simulator has to fulfil many requirements, such as, to be able to simulate hundreds of thousands of various entities at fine grained time granularity (milliseconds) for long periods of time (hours). We here motivate and describe identified requirements and challenges in simulation of the Telco cloud. Addressing the challenges and fulfilling the requirements is crucial while designing a meta-model and implementing a simulator.

Telco cloud stakeholders will benefit from being able to investigate the consequences of possible infrastructure configurations. For example, IPs responsible for building and maintaining the infrastructure may use the simulator for planning placement and capacity for new DCs, as well as modifying capacity and connectivity of existing DCs. Service providers that use infrastructures to host services are interested in comparing different strategies for placement of application components. Moreover, developers that implement mobile applications utilising a Telco cloud, need to determine what application components could benefit from offloading. To answer these and similar questions, tests with various infrastructure configurations need to be performed and results compared. There are two options for performing these tests: by simulation or by running them in real test beds.

Current infrastructure cannot be used for testing the Telco cloud. Creating a physical test bed for large-scale testing of a Telco cloud in different configurations is economically infeasible and small-scale test beds will not be able to capture phenomena occurring in reality, such as, user mobility patterns or latency issues.

For these reasons, we believe that simulation is the most feasible option to evaluate the Telco cloud. However, we have identified several requirements that make simulation of Telco clouds challenging. First of all, the scale of simulation is large in terms of number and types of entities. The simulation of Telco cloud has to concurrently cover hundreds of thousands of MDs moving around a simulated area, each generating requests; hundreds of RBSs providing an access to the core network; and tens of DCs, running services that process requests. Another challenge is the ratio between time precision and length of simulation. We are interested in a very fine-grained latency simulation, that requires precision of at least milliseconds. However, to capture the daily patterns of MD movement (e.g., moving between home, work, and shops) caused by migrations of users carrying them, the whole system needs to simulate several run-time hours. Moreover, a simulation of application statefulness, and of transferring those states when MDs are moving, have not yet been described in the literature and requires new models to be developed.

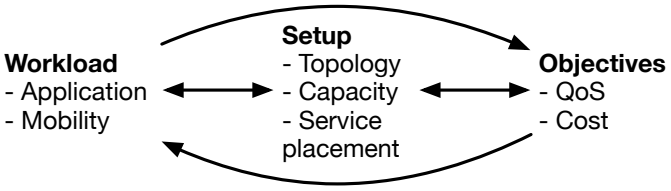


Figure 2: Dependencies between elements of a Telco cloud.

4 Telco cloud Dynamics

Before constructing a meta-model of the Telco cloud we first need to understand Telco cloud fundamental dynamics, construed as the relations between system's input, configuration and output. Later, we will use the knowledge of these dynamics to build the intended simulation meta-model.

Figure 2 visualizes the dynamics inside the Telco cloud. The *workload* is an input to the system that IPs do not have influence over. It includes: applications, with a request generation model (rate and size), a resource requirements model (the amount of resources needed to process a request), and application statefulness (the overhead of transferring user's state between DC); as well as, the mobility of users carrying MDs.

Next, *objectives* describe required output characteristics of the system. We have identified two fundamental objectives. Firstly, QoS, which imposes performance requirements, e.g., latency or throughput through SLO. Secondly, the monetary cost for IP associated with energy consumed for computation and maintenance of an infrastructure. The objectives can be used when constructing an optimisation problem with QoS as conditions and cost as the function that should be minimised.

Finally, *setup* is that part of the system that can be adjusted by designers or operators to achieve desired objectives when processing existing workloads. Setup includes topology, location, capacity of DCs and the network that interconnects MDs and RBSs with DCs, as well as resource management policies that control placement and migration of services.

The above mentioned elements are all dependent on each other. First, the setup of a Telco cloud is related to the existing workload. The capacity of a DC is defined by the resource demands of the services, e.g. how memory-, CPU-, network-, and disk-intensive the services are. The locations and topology of DCs are defined by the geographic and demographic scope of the services, the number of MDs that reside in that domain, and the capacity of the associated telecommunication infrastructure.

Secondly, workload influences objectives. E.g., user mobility is inducing delay during service migration and potentially causes QoS penalties. Moreover, application statefulness introduces additional costs of storing data in a DC and transmitting

data between Proximal DCs. It may also increase latency due to an additional time necessary to send the state data before processing of the request can begin.

Finally, objectives depend on the setup: QoS is proportional to the proximity and capacity of DC – a smaller DC catchment (the geographic area the DC serves) translates to greater locality and reduced propagation latency, while higher capacity allows hosting of more services. Moreover, the capacity and catchment of DCs determine Telco cloud costs. Costs are proportional to dispersion of computing capacity. Firstly, there is an overhead of each DC, irregardless of its capacity, e.g., building, cooling infrastructure, and connection to energy or network. Secondly, dispersion increases costs of maintenance, e.g., technicians have to travel between locations. Therefore, costs are proportionally higher in smaller, dispersed DCs because of high initial costs and proportionally lower in huge, centralized DCs due to the economy of scale [6].

5 Existing models

To support creation of a meta-model that incorporates workload, setup, and objectives of the Telco cloud described in the previous section, we here survey existing models in the following categories: application request generation and resource requirements, MD mobility, networks, DCs, and infrastructure costs.

Most of the models and simulators are assigned to only one of the above mentioned categories, however capabilities of four surveyed simulators extend to many categories, so we summarise them in Table 1.

Table 1: Overview of surveyed simulators.

Framework	RG	RR	M	N	DC
NS-3	✓		✓	✓	
OMNeT++	✓		✓	✓	
CloudSim		✓			✓
GreenCloud		✓			✓

RG – Request Generation, RR – Resource Requirements,
M – Mobility, N – Network, DC – Data Centre.

5.1 Workload Models

Applications running in the Telco cloud consist of a mobile client and a server processing offloaded computations. Therefore, they should be modelled from two perspectives: *request generation* that describes how requests are created and sent to the DCs; and *resource requirements* that describes how much computational resources are needed to process the requests.

Traffic

Traffic models capture the user's behaviour by primarily representing interaction times or the timing clicks through a stochastic process, often Poissonian in nature. A user behavioural model can be further refined by introducing a stochastic model for how long time a user consumes a certain type of content. Additionally, the transition between types of content is often modelled as a Markov process.

Furthermore, the traffic characteristics are commonly modelled with multiple stochastic processes, encompassing the number of packets in a session, and the size of each packet. Traffic models are either closed or open looped. In an open loop model, the generation of each new session is typically a Poisson process independent of the resulting DC action. Conversely, in a closed loop model, the generation of new sessions is dependent on timing of the response from the DC and thus the properties of the previously generated session.

In the packet-level event driven network simulators, NS-3 [7] and OMNeT++ [8], a node can act as either a client or server, by the mechanism of either sending packets provided by a stochastic model, at a given rate, within a certain time period, and at a certain interval, or processing received packets from a buffer, at a given rate. Both server and client models can be augmented with a more complex system of queues to such an extent that they can represent an abstract DC that hosts multiple applications.

Resource Requirements

CloudSim [9], which is a simulator of cloud infrastructure, provides an application model that describes computational requirements – the amount of resources that needs to be available (e.g. number of cores, memory and storage); and communicational requirements – the amount of data that needs to be transferred. GreenCloud [10], which is a packet level simulator based on NS-2, apart from computational and communicational requirements, describes also QoS requirements, expressed by an execution deadline. The application model may also include the size of the code that has to be offloaded and dependencies on other services, e.g., in terms of amount of data that has to be sent or received [11].

Mobility

The NS-3 and OMNeT++ nodes described above can be set into motion given a certain stochastic mobility model. They can for example traverse the space as pedestrians, or auto mobiles, with corresponding velocity and rate of change. The spatial relationship between nodes and RBS affects the prevailing channel properties and RBS-to-node associations. Node mobility will also result in handover between RBSs, which in turn will alter the paths of the node-generated workload in the network.

5.2 Setup Models

Below, we describe existing models and simulators of networks and DCs, which can be used to configure the setup of the Telco cloud meta-model.

Network

There are several well-established event-driven frameworks that are capable of modelling computer networks, mobile networks, applications, packet-level network traffic, infrastructure, and independent users. The two primary examples are, mentioned in the previous section, NS-3 and OMNeT++. These two are commonly deployed in academic network research and provide detailed results on network utilisation, throughput, congestion, and latency.

Both NS-3 and OMNeT++ are comprehensive packet-level network simulation frameworks that include wired and wireless standards, and are able to simulate communication channel conditions. Furthermore, both frameworks have detailed models for channel definition, such as propagation delay, interference, data rate, and medium access schemes. In addition, to a varying degree, NS-3 and OMNeT++, by default or through extension, support control plane signalling for a number of wireless standards and complex network topologies.

Both frameworks have support for modelling different types of network nodes, ranging from computers to routers and switches. Each edge and node pair has a defined communication and medium access standard, such as TCP/IP and Ethernet. Each packet that is sent over the network is treated in accordance with the prevailing network and transport protocols and routing standard. In both, the event of arrival and departure of packets drives the simulation clock.

Furthermore, they require detailed configuration of all communication modes as well as node behaviour, making it very time-consuming to implement and verify systems with different levels of abstractions, and are thus cumbersome to model abstract systems.

The Telco cloud topology is yet to be defined with unspecified control planes, it would thus be counter-intuitive and time consuming to implement Telco cloud topologies in either NS-3 or OMNeT++. In some instances, some modules would have to be completely redesigned, and others would have to be specified to a much greater detail than the Telco cloud can offer at this stage.

Data Centre

The purpose of this section is to survey the DC models that are the most suitable for inclusion in the Telco cloud meta-model. An extensive list of mathematical models, simulation approaches, and test beds can be found in [12], while [13] provides a survey

of twelve cloud simulators. After careful examination, we chose the ones that best suit our goals.

We compare DC models and simulators based on descriptions provided by the authors of the simulators. For each model we describe: *Resource Provisioning* – what resources are included and how they are modelled; *QoS* – what performance indicators are measured; *Costs* of computation in the DC; *Performance* of simulator – an estimation of the time needed to perform a simulation.

CloudSim is an event-based simulator implemented in Java, for simulation of cloud computing system and application provisioning environments.
Resource Provisioning. The CloudSim simulation layer offers dedicated management interfaces for CPU, memory, storage and bandwidth allocation, as well as, defining policies in allocating hosts to VM – VM provisioning. Hosts are described by processing capabilities (in MIPS) and a core provisioning policy, together with an amount of available memory and storage. A model supports time-sharing and space-sharing core provisioning policies on both host and VM levels.
Latency (QoS). The latency model is based on conceptual networking abstraction, where the communication delays between each pair of entity type (e.g. host, storage, end-user) are described in a latency matrix as a constant value expressed in simulation time units (e.g. milliseconds).
Costs. CloudSim provides a two-layered cost model, where the first layer relates to Infrastructure as a Service (IaaS), with costs per unit of resources, while the second one relates to Software as a Service (SaaS), with costs per task units (application requests). This model allows calculation of the costs of using the cloud from the end-user perspective or the revenue from the IP perspective.
Performance. CloudSim is able to perform large-scale simulations, e.g., it can instantiate an experiment with 1 million hosts in 12 seconds. Moreover, memory usage grows linearly with the host number and even with 1 million hosts it does not exceed 320 MB.

CloudAnalyst [14] is a simulator of geographically distributed large-scale cloud applications, developed with Java and that utilises CloudSim and SimJava.
Resource Provisioning. Cloud Analyst uses the same resource provisioning model as CloudSim.
Latency (QoS). A latency model allows configuration of network delays, available bandwidth between regions, and current traffic levels. CloudAnalyst facilitates experiments with latency by producing following statistical metrics: average, minimum, and maximum response time of all user requests; and response time grouped by time of the day, location, and DC.
Costs. CloudAnalyst supports calculation of costs for using cloud resources, such as cost per VM per hour and cost per Gigabit of data transfer.
Performance. To improve performance of simulation entities are grouped at three levels: clusters of users, cluster of requests generated by users, and clusters of requests

processed by VM.

GreenCloud is a packet level simulator based on NS-2, for simulation of energy-aware clouds.
Resource Provisioning. Servers are modelled as a single core node with defined processing power limit (in MIPS or FLOPS), size of memory and storage, and implementing different task scheduling mechanisms.
Latency (QoS). Full support for the TCP/IP protocol reference model is provided and thanks to that the simulator is able to calculate communication latency with high accuracy.
Costs. GreenCloud allows detailed modelling of energy consumption by implementing energy models for every DC element.
Performance. Given that GreenCloud has to simulate the full stack of Internet protocols, each simulation only takes in the order of tens of minutes for a DC with a few thousands of nodes.

5.3 Costs Models

The above mentioned DC models focus mostly on the costs of running applications in DCs from the end-user perspective. Since we want to model costs of DCs from the IP point of view, additional models are needed for capital expenditures (CAPEX) and operating expenditures (OPEX).
CAPEX includes costs of infrastructure that needs to be built and servers that have to be bought.
Infrastructure Costs. Costs of building, power distribution, (and cooling can be estimated using a following equation: $\$200M \cdot (1+c_m)/a_i$, where c_m is the cost of money[2], and a_i is the time of infrastructure amortisation [in years] [15].
Server Costs. Costs of servers can be modelled as $n_s \cdot p_s \cdot (1+c_m)/a_s$, where n_s is the number of servers, p_s is the price of one server [in \$], c_m is the cost of money, and a_s is the time of server amortisation [in years] [15].
OPEX consists of power and personnel costs.
Power Costs. To estimate costs of power, the following equation can be used, $n_s \cdot pc_s/1000 \cdot PUE \cdot p_{KWH} \cdot 24 \cdot 365$, where n_s is the number of servers, pc_s is the power consumption of one server [in W], PUE is Power Usage Efficiency, and p_{KWH} is the price of electricity [in \$ per KWH] [15].
Personnel Costs. Costs of personnel can be calculated using $M_1 \cdot C_1 + M_2 \cdot C_2 + M_3 \cdot C_3$, where M_1 is the number of IT personnel per rack, M_2 is the number of facility personnel per rack, M_3 is the number of administrative personnel per rack, and C_1, C_2, C_3 are the average costs per person for each of the above mentioned categories [16].

[2]Cost of money is the rate of interest or dividend payment on borrowed capital.

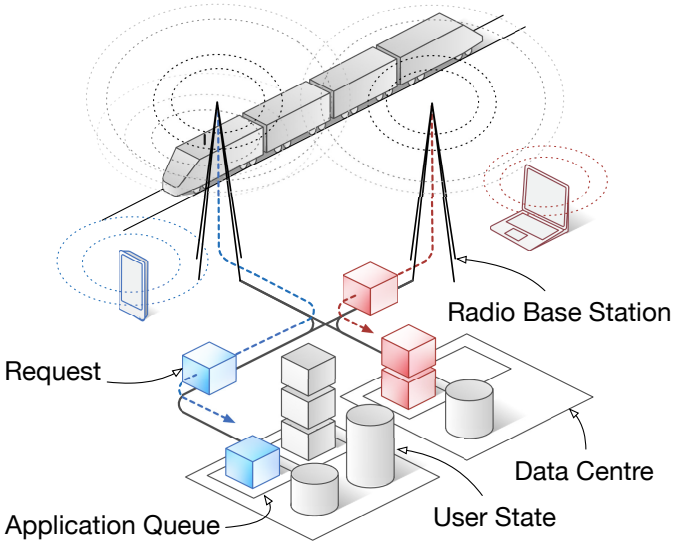


Figure 3: Visualisation of Telco cloud meta-model.

6 Telco cloud Meta-model

We here detail how we have composed the above surveyed models into a Telco cloud meta-model. Figure 3 depicts the visualisation of the meta-model. MDs, such as cell phones or laptops, are carried by end-users, who are in motion. The MDs generate requests which are sent over the network to a DC. It is also possible that requests are generated by sensors that may be static (e.g. traffic cameras) or mobile (e.g. trains). The requests are processed in the DC and the response is sent back to the MD or sensor. Processing requests, in case of statefull applications, generates a user state, that has to be migrated with the end-user if he moves to another DC.

The primary objective of the meta-model is to capture the interactions between application workload, MD mobility, network topology, and DC characteristics, and their influence on QoS and costs of Telco cloud. The parameters that define the meta-model are presented in Table 2 and described in detail below.

6.1 Workload Model

The first group of parameters in Table 2 describes the mobility of end-users carrying MD and the characteristics of requests generated by these MDs.

Table 2: Fundamental meta-model parameters.

Type	Parameters	Unit	Description
WORKLOAD			
Request generation	N_{ser}		Total number of services
	λ^i_{ses}, where $i = 1,2,\ldots,N_{ser}$	s	Session arrival rate to DC
	N^i_{req}, where $i = 1,2,\ldots,N_{ser}$		Number of requests per user session
	S^i_{req}, where $i = 1,2,\ldots,N_{ser}$	KB	Size of requests for a given service
	D^i_{req}, where $i = 1,2,\ldots,N_{ser}$	s	Inter-request time
Resource Requirements	CPU^i_{idle}, CPU^i_{req}, where $i = 1,2,\ldots,N_{ser}$	MI	CPU cycles used by service
	mem^i_{idle}, mem^i_{req}, where $i = 1,2,\ldots,N_{ser}$	MB	Size of memory used by service
	$disk^i_{idle}$, $disk^i_{req}$, where $i = 1,2,\ldots,N_{ser}$	MB	Size of storage used by service
	$state^i$, where $i = 1,2,\ldots,N_{ser}$	MB	Size of user's state produced per req.
Mobility	N_{MD}		Number of Mobile Devices
	s^i_t, a^i_t, θ^i_t, ω^i_t, where $i = 1,2,\ldots,N_{MD}$		Movements of Mobile Devices
SETUP			
Network	N_{RBS}		Number of Radio Base Stations
	d_{RBS}	m	Dimensions of an RBS cell
	D_{net}	s	Cumulative network delay
Data Centre	N_{DC}		Number of Data Centres
	N^i_S, where $i = 1,2,\ldots,N_{DC}$		Number of servers in Data Centre
	N^j_{CPU}, where $j = 1,2,\ldots,N^i_S$		Number of CPUs per server
	s^j_{CPU}, where $j = 1,2,\ldots,N^i_S$	MIPS	CPU's speed
	$memory^j$, where $j = 1,2,\ldots,N^i_S$	MB	Amount of memory per server
	$storage^j$, where $j = 1,2,\ldots,N^i_S$	GB	Amount of storage per server
	$network^i_{bw}$, where $i = 1,2,\ldots,N_{DC}$	Mb/s	Network bandwidth
	t_{init}, t_{idle}, t_{term}	s	Times of VM transitions
Service Placement	$placement = \{every, n\text{-}closests\}$		Service placement policy
OBJECTIVES			
Quality of Service	RT^i, where $i = 1,2,\ldots,N_{ser}$	s	Application response time
	TP^i, where $i = 1,2,\ldots,N_{ser}$	req/s	Application throughput
Costs	$Cost$	$	Total costs of infrastructure

Request Generation

Many services may run in the Telco cloud at the same time and their number is defined by N_{ser}. We model a service application as a stateful web service. Each session is separated in time with a Poisson process λ_{ses} [17]. Each session produces N_{req} requests, sampled from an inverse Gaussian distribution, where each request is separated in time by Log-Normal distributed delay D_{req} in seconds. The size of each request is given by S_{req} KB and is drawn from a Pareto distribution.

Resource Requirements

To model application resource requirements we propose a linear model specifying the needed amount of resources, both for an idle service and per processing each request. An idle service uses CPU_{idle} CPU operations, mem_{idle} amount of memory, and $disk_{idle}$ amount of storage. Additionally for each processed request, the service

uses CPU_{req} CPU operations, mem_{req} amount of memory, and $disk_{req}$ amount of storage. The amount of user's state data created by each request is defined by $state$ and expressed in absolute value or percentage of request size S_{req}.

Mobility

The network is populated by N_{MD} MDs, each subscribing to a subset of the N_{ser} available services. The 2-dimensional, multi modal, mobility model detailed in [18] provides us with an on-average uniform distribution of users, with movement proportional to the duration of a session and the scale of the mobile network. The aforementioned model defines the properties of an MD's movement. A MD's momentary movement is defined by its velocity constituted by the current speed s and current direction θ. Changes in mobility are defined by multiple stochastic processes that describe the duration of its state. An entity's speed s is independent of direction θ and is maintained for T_s seconds, after which acceleration a is applied between a_{min} and a_{max} for time T_a, until it reaches s_{min} or s_{max}. Furthermore, direction θ is maintained for time T_θ until the next change-event where the direction θ is altered for T_ω seconds with at the rate of ω radians per second. T_s, T_a, T_θ, and T_ω, describing the timing of each change-event, are set for each mobility mode, and are each defined by a probability distribution bounded by a maxima and minima.

6.2 Setup Model

The second group of parameters in Table 2 characterises the network and DCs.

Network

In our model, the core network introduces a cumulative propagation, switching, and routing delay and it is modelled with a Weibull delay D_{net} in multiples of the number of network nodes between the source and the destination [19]. The network distance between RBSs is equal to the cell dimension d_{RBS}. The associated RBSs are equidistant to their common DC, and are for the sake of simplicity assumed to be separated by one network edge.

Furthermore, forthcoming cell planing practices aim to increase area energy efficiency by favouring smaller cells in urban areas [20, 21]. Our model employs a small homogeneous mobile network composed of N_{RBS} equidistantly distributed RBSs.

In the absence of a specific mobile generational standard, an MD is handed over between RBSs at the geographic point where they cross the cell boundary distinguishing two independent RBSs defined by the width of the rectangular cells d_{RBS}.

Data Centre

The DC model captures the influence that its capacity has on performance and costs of computation, as described in Section 4.

To capture the influence on performance, quantity and quality of each DC resource is described. DC consists of N_S servers, that can differ in specification. Server contains N_{CPU} CPUs capable of executing s_{CPU} operations in every second. Values of *memory* and $storage$ specify the total amount of available memory and storage, respectively. The network bandwidth is specified with $network_{bw}$. The DC model includes also a provisioning model, that describes how available resources are shared among several applications, e.g., time-sharing or space-sharing.

A DC hosts services in VMs. A service can be distributed over multiple VMs. Incoming workload is load-balanced by either a method of round-robin, random selection, or placed in the VM with the lowest load. However, a user requests are always forwarded to the VM that served his first request. A service can specify a minimum and maximum number of VMs it requires. The DC scales the application within these bounds based on the load-balancing outcome.

To emulate the life-cycle of a VM we have defined six VM states described in Table 3. The transitions between the states are presented in Figure 4. At the beginning all VMs are in INACTIVE state. A VM is initiated when the first request arrives to a DC. It takes t_{init} seconds before VM is ready to start processing requests or receiving migrated requests and user state from other DC. We assume that a VM is not able to process requests and handle migrations at the same time, so it changes state between PROCESSING and MIGRATION over the time. Moreover, we give migrations a higher priority than processing, so processing is paused if there are any migrations to perform. When there are no requests to process and no migrations to handle a VM goes into IDLE state. A VM is terminated if IDLE state lasts for longer than t_{idle} seconds, and the VM termination takes t_{term} seconds.

Table 3: States of Virtual Machine.

Name	Description
INACTIVE	VM is turned off.
INITIATING	VM is booting up.
PROCESSING	VM is serving requests.
IDLE	VM is waiting for requests.
MIGRATING	VM is transmitting data.
TERMINATING	VM is shutting down.

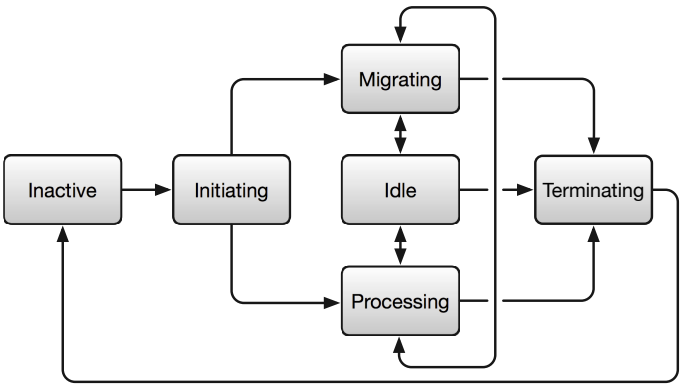


Figure 4: Transitions between Virtual Machine states.

Service Placement

Service placement policies define in what DC(s) a service should be hosted, what number of replicas should be running, and when a service should be migrated between DCs. These decisions depend on the mobility of users, the size of users' state that has to be migrated, and QoS requirements. For example, a service can be hosted in n Proximal DCs closest to the majority of its users (*n-closests*), or in the case of latency sensitive services in every Proximal DC that is needed to provide acceptable QoS (*every*).

6.3 Objectives Model

The third group of parameters in Table 2 describes QoS and costs of the Telco cloud.

Quality of Service

Combining the resource requirements model, which describes the amount of resources an application needs, with a DC model, allows to simulate how collocation of different services in a DC influences their response times RT^i and throughputs TP^i.

Costs

In our opinion the cost models available in the literature and described in Section 5.3 are very "country dependent", because of the inclusion of variable parameters such as salaries, costs of energy or costs of property. They are also not taking into account parameters important from the perspective of the Telco cloud, such as the size of DC.

Therefore, we model the costs of the Telco cloud using a basic heuristic based on observation that dispersion of infrastructure causes additional costs, e.g.: increase of administrator travel time between locations, and higher unit costs of computation in proximal DCs because of smaller scale and high initial costs.

$$\text{Cost} \propto \frac{N_{\text{DC}}}{\sum^{\text{DC}} N_S} \tag{1}$$

As shown in Equation 1, the total cost of a Telco cloud is directly proportional to the number of DCs and inversely proportional to the total number of servers in all DCs. It means that distributing the same number of servers among many DCs is more expensive than placing them in one DC.

6.4 Limitations

The proposed meta-model has several limitations. The application model assumes that all requests generated by one application are homogeneous and each of them consumes the same amount of resources. The mobile access network model does not take into account the physical layer, channel provisioning, and cell load balancing. Additionally, the radio access network functions as a mechanism to associate MDs with DCs propagation and system processing delays are thus not modelled.

7 Simulation showcase

We have implemented a coarse grained simulator using SimJava [22] as the underlying event-driven simulation framework. All modules are implemented from scratch but are based on the meta-model presented in Section 6. The simulator fully implements the proposed request generation and network models, but has implemented more abstract mobility, resource requirements, DC, and service placement models.

To demonstrate the scope of the Telco cloud meta-model and the simulator we introduce an elementary showcase scenario below. The scenario is designed to reveal the basic relationship between workload – MD mobility, setup – Proximal DC catchment, and objectives – the aggregate utilisation of a Telco cloud.

7.1 Experiments

For the sake of clarity we present a simplified scenario. Only one service in considered and the size of simulation is reduced when compared to the desired scale. The VM scalability and placement models are just a proof-of-concept. The goal is to obtain clear conclusions about the relation between MD mobility and DC catchment, and avoid the interference of other elements. The scenario is described in detail below.

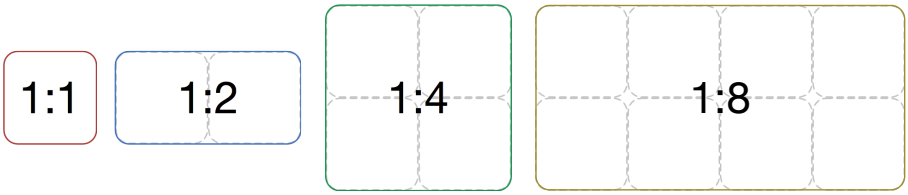


Figure 5: RBSs ranges and DC catchments.

The telecommunication infrastructure is composed of 16 RBSs, in a 4x4 layout, as presented in Figure 5. The cells, depicted with dashed lines, are tangent but not overlapping and are dimensioned as a typical LTE micro-cell at 750 m, as detailed in [20]. The number of DCs varies between the experiments and thus so, also the DC catchment, represented with the solid lines, and defined as the ratio between DCs and RBSs, changes between (1:1) and (1:16). Note that (1:16) catchment covers whole simulation area with one DC. In abstract terms, the (1:1) catchment represents a setup with one Proximal DCs per RBS. In contrast, the (1:16) catchment approaches a more traditional case of Remote DC serving all users in the domain.

To reveal the effects of DC catchment, all DCs are of the same capacity. The number of VMs in each DC is scaled proportionally to the number of users they serve. The DC in the (1:16) catchment scenario has 16 VMs, while the DC in the (1:1) scenario has just one VM. The workload is balanced among available VMs, new sessions are forwarded to the least loaded VM. To reveal the full extent of the effect of user mobility, user states and requests are strictly migrated to the geographically nearest DC.

We use a request generation model with a session arrival rate of λ_{ses} described by a Log-Normal distribution with the parameters $\mu = 3$ and $\sigma = 1.1$. The number of requests per session N_{req} is taken from an Inverse Gaussian distribution with the parameters $\lambda = 5$ and $\mu = 3$. Inter-request time is D_{req} seconds and is modelled with with an Exponential distribution with $\lambda = 0.1$. The simulation domain is populated by 480 MDs, all subscribing to the same service. Due to the size and simplicity of the network topology in the proposed scenario, we are deploying a Markov-based mobility model. The mobility mode is based on a car and is as specified in Section 6.1, with parameters from [18]. To allow the mobility and workload models to jointly reach a steady state, the simulation is run for 8 simulated hours. This results in an average processing load of 30%, this level should give enough margin to for example migrations to complete successfully.

The user state is proportional to the aggregate size of that user's sessions with the application it subscribes to and is defined by a 5^{th} order AR-process with linearly de-

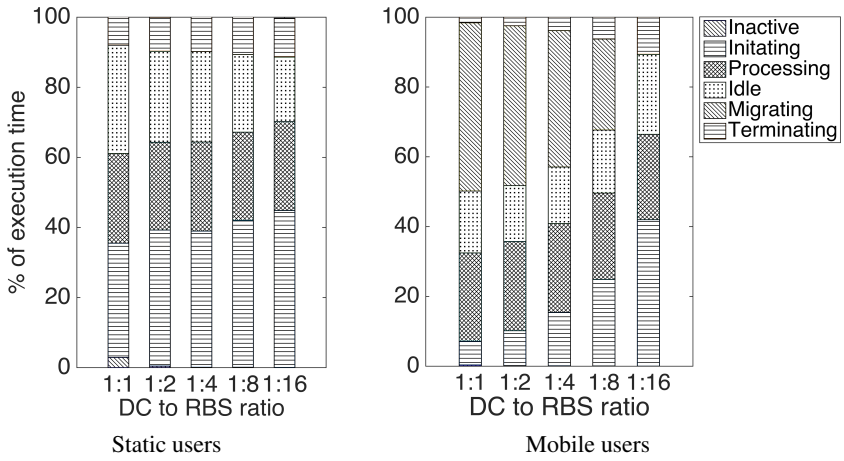


Figure 7: DC catchment vs. time spent in each VM state.

caying parameters. In our simulation, initialisation of a VM takes $t_{\text{init}} = 81s$, similarly as for m1.small VM type in Amazon EC2 [23]. A VM is terminated if it remains in the IDLE state longer than t_{idle} which is equal to the mean inter-session time. It takes $t_{\text{term}} = 21s$ to terminate a VM.

In order to investigate the influence of Proximal DC catchment on the aggregate performance of the Telco cloud we observe the life cycle of the VMs that run within the DCs by recording the amount of time each VM spends in each state. We run two sets of experiments. In the first set, end users are static. The second set of tests introduces mobility. In both sets we investigate the variations in the distribution of time that VMs spend in each state.

7.2 Results

Figure 7 shows the breakdown of the mean time spent in each VM state in the system per DC catchment. With a (1:1) DC catchment the utilisation suffers from the proportion of time spent in IDLE state due to the relatively low request arrival rate generated by one sixteenth of all users. The inefficiency is caused by the time the system spends in the IDLE, INITIATING, and TERMINATING states. The composition of time spent in these states changes with DC catchment, and is a reflection of the number of VMs in a DC and load-balancing effort. Reducing the time spent on starting and terminating VMs would free up more resources and perhaps also make the system more reactive to sudden workload changes. The intelligent management of VM scalability and placement is clearly something that needs to be optimised.

Figure 6b reveals the overhead of user mobility and the migration effort it incurs. Depending on the DC catchment, different migration dynamics come into play. As migrations are more frequent in the (1:1) case than in the (1:8) case, user states do not have the time to grow as much between migrations in the former case. The migration effort is therefore not a factor eight lower in the (1:8) case versus the (1:1) case, but rather, they spend 26% and 47% of their time in the MIGRATING state, respectively. The system dynamics revealed by Figure 6b, where at worst, 47% of the execution time is spent migrating users, points to the need to find scaling mechanisms for the Telco cloud that take into account mobility and inactivity, so that resources can be freed dynamically for other revenue generating applications. A policy of strictly migrating user states and requests to the geographically closest DC, irregardless of DC catchment, in order to obtain minimal propagation and communication latency, is suboptimal.

8 Conclusions

In this paper we present a way to combine existing models of user mobility, mobile and core networks, and DCs into a meta-model capable of capturing dynamics of the Telco cloud. We also implement a prototype simulator based on a simplified meta-model.

The meta-model can be used by telecommunication operators as well as equipment developers to model existing infrastructures and to plan future changes. Researchers can test algorithms for resource management, e.g., migration of services between geo-distributed DCs. Also developers can benefit from using the simulator to observe how their mobile applications behave in Telco cloud environment.

Future work will be focused on enhancing the functionality of the simulator to incorporate other parameters from the presented meta-model. Then, using the simulator we would like to explore the following Telco cloud challenges: minimising the trade-offs between costs and performance of Telco cloud depending on the DC placement and capacity, and optimal placement and migration of services between DCs.

Acknowledgements

This work is funded by the Swedish Research Council (VR) project Cloud Control and the European Union's Seventh Framework Programme under grant agreement 610711 (CACTOS). Maria Kihl and William Tärneberg are members of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council. Also, they are members of the Excellence Center Linköping – Lund in Information Technology (ELLIIT).

We thank Johan Eker (Ericsson Research) who contributed with feedback during several discussions and Tania Lorido-Botran (University of the Basque Country) for her critical comments on early drafts of the paper.

References

[1] P. Bosch, A. Duminuco, F. Pianese, and T. L. Wood, "Telco clouds and virtual telco: Consolidation, convergence, and beyond," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 982–988.

[2] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. ACM, 2010, pp. 35–46.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *Journal of Lightwave Technology*, vol. 31, no. 4, pp. 523–537, 2013.

[5] F. Baroncelli, B. Martini, and P. Castoldi, "Network virtualization for cloud computing," *annals of telecommunications-annales des télécommunications*, vol. 65, no. 11-12, pp. 713–721, 2010.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[7] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

[8] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM'2001)*, vol. 9, no. S 185, 2001, p. 65.

[9] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[10] D. Kliazovich, P. Bouvry, and S. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11227-010-0504-1

[11] D. Kovachev, "Framework for computation offloading in mobile cloud computing," *Int. J. of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 7, pp. 6–15, 2012.

[12] G. Sakellari and G. Loukas, "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing," *Simulation Modelling Practice and Theory*, vol. 39, no. 0, pp. 92 – 103, 2013, s.I.Energy efficiency in grids and clouds. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X13000658

[13] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 866–872.

[14] B. Wickremasinghe, R. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, April 2010, pp. 446–452.

[15] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[16] C. D. Patel and A. J. Shah, "Cost model for planning, development and operation of a data center," 2005.

[17] A. Reyes-Lecuona, E. González-Parada, E. Casilari, J. Casasola, and A. Diaz-Estrella, "A page-oriented www traffic model for wireless system simulations," in *Proceedings ITC*, vol. 16, 1999, pp. 1271–1280.

[18] C. Bettstetter, "Smooth is better than sharp: A random mobility model for simulation of wireless networks," in *Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile*

Systems, ser. MSWIM '01. New York, NY, USA: ACM, 2001, pp. 19–27. [Online]. Available: http://doi.acm.org/10.1145/381591.381600

[19] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an ip backbone network," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 6, pp. 908–921, 2003.

[20] S. Shahab, T. Kiong, and A. Abdulkafi, "A Framework for Energy Efficiency Evaluation of LTE Network in Urban, Suburban and Rural Areas," *Australian J. of Basic and Applied Sciences*, vol. 7, no. 7, pp. 404–413, 2013.

[21] A. Fehske, F. Richter, and G. Fettweis, "Energy efficiency improvements through micro sites in cellular mobile radio networks," in *GLOBECOM Workshops, 2009 IEEE*, Nov 2009, pp. 1–5.

[22] F. Howell and R. McNab, "Simjava: A discrete event simulation library for java," *Simulation Series*, vol. 30, pp. 51–56, 1998.

[23] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing." Springer, 2010, pp. 115–131.

Paper III

Resource Management Challenges for the Infinite Cloud

Cloud applications are growing more and more complex because of user mobility, hardware heterogeneity, and multi-component nature. Today's cloud infrastructure paradigm, based on distant data centers are not able to provide consistent performance and low enough communication latency for future applications. These discrepancies can be accommodated using existing large-scale distributed cloud infrastructure, also known as Infinite Cloud, which is amalgam of several Data Centres hosted by a Telecom Network. The Infinite Cloud provides opportunity for applications with high capacity, high availability, and low latency requirements. The Infinite Cloud and federated cloud paradigms introduce several challenges due to the heterogeneous nature of the resources of different scale, latencies due to geographical locations and dynamic workload, to better accommodate distributed applications with increased diversity. Managing a vast heterogeneous infrastructure of this nature can not be done manually. Autonomous, distributed, collaborative, and self-configuring systems need to be developed to manage the resources of the Infinite Cloud in order to meet application Service Level Agreements (SLAs), and the operators' internal management objectives. In this paper, we discuss some of the associated research challenges for such a system by formulating an optimization problem based on its constituent cost models. The decision maker takes into account the computational complexity as well as stability of the optimal solution.

William Tärneberg, Amardeep Mehta, Johan Tordsson, Maria Kihl, Erik Elmroth,
"Resource Management Challenges for the Infinite Cloud,"
Feedback Computing Workshop at CPSWeek 2015. 2015), Seattle, United States, April, 2015.

1 Introduction

As cloud computing is transforming the applications, usage patterns, and business models of today, its realization has not yet achieved its full potential. For cloud resources to be ubiquitous and to truly offer computing as a utility, future infrastructure generations will need to be capable of meeting the above expectations. New applications having different run times will be highly distributed, run on heterogeneous hardware and software with low latency and high availability requirements. The Infinite Cloud's end-users should be agnostic to where and how their application or content is stored and executed, irrespective of its complexity and size. The cloud capacity-abyss should seemingly, without doubt, absorb whatever is submitted to it. One way to realize this goal and to accommodate the increased plurality of applications, is to augment and diversify the existing cloud capacity beyond infinity through federated clouds and telco networks. Through its distributed cloud resources in a telco's network, an Infinite Cloud introduces an decreased latency and compute capacity diversity, and enables network aware applications. In union with the federated cloud paradigm, more diverse sets of resources can be offered and brokered to any cloud application, specific to where its users are, see Figure 1.

With increased resource plurality, new types of applications, and greater expectations on cloud services also comes new challenges. The vast number of heterogenous resources will need to be autonomically managed with feedback from both external and internal inputs for efficient resource utilisation.The autonomous systems should collaborate holistically to minimise global energy, compute, and network resource usage [1].

2 Infinite cloud motivation

Distributed application execution is becoming more seamless, to match their individual performance and latency requirements, different tiers are now executed in geographically distributed DCs, often structured as federated cloud [2]. Cloud applications are at an increasing rate being accessed from MDs. The boundary between discrete application components and how and when they interact with the users and objects in their vicinity is becoming increasingly opaque. Cloud applications such as those with dynamically generated content and critical control process require today unattainably low communication latency. The geographic separation between end-user and the DC in which the application is hosted introduces unwanted communication delay and jitter. Moreover, due to intermittent cloud capacity availability and latency inconsistencies, the current prevailing smartphone app paradigm resorts to executing the majority of an application locally in the MD, as opposed to in the cloud [3].

MDs have limited and scarce compute and reserved energy capacity. MD vendors strive to create an experience of perceived desktop-class compute performance and an infinite energy reserve. This sought-after experience can be achieved through application offloading using the available resources in the Infinite Cloud [4]. Offloading can generally be deemed worthwhile if the available compute capacity of a DC exceeds that of the MD and the energy consumed executing the application on the MD exceeds the energy consumed communicating application interactions, graphics changes, and user states, provided that it can be done with a low enough communication latency [5]. The Infinite Cloud provides the necessary infrastructure to offload applications from MDs to proximal DCs where compute resources and power is cheap and is accessible with a low communication latency, [6]. Offloading can either come in the form of remote code execution, or in the form of more immersive cloud applications that behave like smartphone apps. In the Infinite Cloud, residing applications can be made network-aware by granting them access to information about the state of the entire network and the paths to the users it serves.

With the emergence of Internet of Things (IoT), a large number of devices are being interconnected and connected to the Internet, at a rapid rate [7]. These devices, ranging from keyholes, to flower pots, to luggage, aggregatly produce and receive a huge amount of data. Between a fair number of these devices exists an ad-hoc and circumstantial sensor to actuator relationship. The sensors often lack any coordination in-between sensors based on their functionality and location. As a result, the data is highly contextually correlated or even redundant. Instead of transporting all that data to a distant cloud DC for analysis, real-time event stream processing at the edge of the network can distill the raw data to dismiss redundant and unwanted information. Similarly, autonomously gathered contextual information about the surrounding environment from multiple proximally located Augmented Reality devices could find it beneficial to share and process that information collectively, in the Infinite Cloud.

3 Challenges

The highly distributed and heterogeneous nature of the Infinite Cloud introduces several interesting autonomous resource management challenges arising from a highly dynamic workload, heterogeneous resources, rapid user mobility, heterogeneous energy costs, and multi-component applications, [8]. The holistic objectives of an Infinite Cloud is to ensure that it persistently meets the SLAs of the applications it hosts, while minimising its total resource usage, including energy. It should do so proactively by dynamically placing and scaling applications primarily by means of feedback input of prevailing foreground network traffic, DC utilisation levels, and application workload and user location changes, see Figure 1.

One of the foremost challenges in the Infinite Cloud paradigm is how to manage

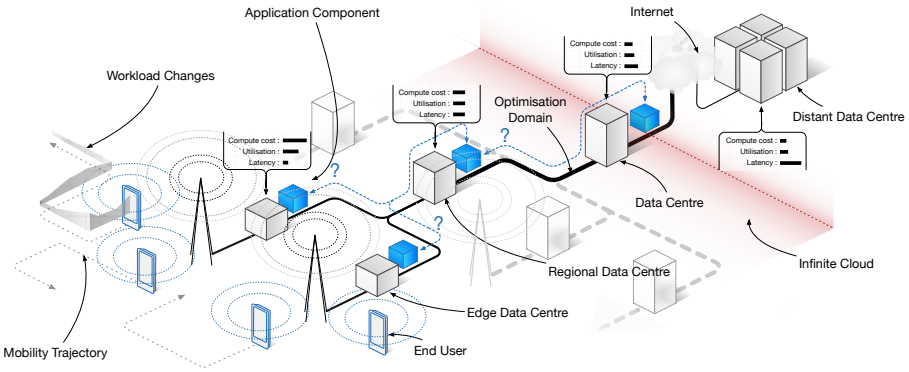


Figure 1: Placement in the Infinite Cloud topology

the highly heterogeneous and distributed resources in a complex system. The sheer size of the infrastructure and the number for management parameters renders a fully centralized resource allocation strategy infeasible [9]. As a result, a decentralised collaborative resource management approach needs to be considered. When reevaluating an application placement decision the systems needs to determine if the energy, compute, network, and latency cost fall short of any of the possible placement possibilities that qualify, for a certain period of time. The systems rate of change determines the duration under which a decision is valid. The number of possible placement combinations and the rapid rate of change means discrete placement decisions need to rely on workload, resource availability, and user location prediction. The system needs to distributedly and collaboratively re-evalute the placement of application components, whenever workload changes for an application, when new applications arrive or are terminated, when applications scale up or down, or when foreground traffic volumes change. The triggers and decisions need to be at the granularity of individual application components, thus arguably distributed. The propagation of the collaborative efforts will thus be bounded by the network's topology and of each application's possible placement alternatives.

The management of complex distributed systems is not trivial [10]. If one placement decision fails to find a solution or is sub-optimal, the effect can have reproductions throughout the system, leading to sub-optimal decisions by peer controllers. Self-oscillations need to be mitigated through feedback control [11] that account for the performance of all of its collaborative peers. The challenge is to construct an autonomous distributed resource management system that is able to self-mitigate and self-heal from individual application, DC, network, and system failures. In addition to internal self-inflicted threats, the systems autonomous components also need be able

to collaboratively contain and eliminate security threats emanating from both within and beyond the Infinite Cloud.

4 Approach

The topology depicted in Figure 1 reflects the union of a Mobile Network Operators (MNOs) network and a federated cloud infrastructure and should be seen as an abstraction of the Infinite Cloud topology proposed in [6]. The placement and scale of the Infinite Cloud DCs will be heavily dictated by the degree of an MNOs infrastructure virtualisation [12], the degree of convergence of core and access networks, and the prevailing geographic demand for cloud services. Although some bounds can be constructed, these properties are not yet defined as the design of forthcoming mobile access network standards and topologies are far from being finalised [13].

Existing 3rd and 4th generation mobile access networks are prevailingly tree-structured [14]. This general structure will feasibly be replicated in future mobile infrastructure generations. Furthermore, the bandwidth availability increases towards a root/source of the network and circadian traffic patterns vary with distance to the source nodes. Similarly, communication latency and jitter will decrease with increased proximity between the DC hosting the application and the application's end-users.

Various MNO IT infrastructure nodes, such as, regional and provincial office facilities and infrastructure hubs are spread throughout the tree's nodes. It is in this existing infrastructure that the Infinite Cloud will proposedly be hosted [6]. The compute capacity is feasibly proportional to the aggregate number of users that can access a DC, successively decreasing towards the tree's endpoints. However, given the lack of research into the scale of these DCs, there is very little we are able to specify to this effect. Operational compute cost on the other hand will arguably be proportional to the distance to one of the trees roots, increasing towards the network's end-nodes.

The communication latency of an application is dictated and maintained by the application's relative locations to its subset of *end users* and the level of congestion on the links it employs. As the *end-user* mobility from one edge node to another can be highly dynamic, the size and location of applications population of *end users* can vary with time. An application's up- and down-link bandwidth is bounded by the available bandwidth which is shared with time-variant *foreground traffic*, that is assumed to be prioritised in a Telecom network.

A MNOs infrastructure resources are finite, and its objectives are to globally meet the Service Level Objectives of each application and the network's foreground traffic, while minimising its Operational Expenditure (OPEX) and bandwidth usage. Each DC imposes an operational *compute cost*, it is assumed that operational compute cost at the edge nodes of the network is always equal to or greater than to those at the entry points of network. Similarly *bandwidth* cost varies with distance to the edge of

the network. An application incurs an up- and down-link *bandwidth* intensity, has a certain *storage* intensity, and *compute* intensity.

Our approach is to design a scheduler that maintains operational stability and minimizes the overall cost for initial deployment and placement through continuous migration of application components on physical machines, intra-DC and/or inter-DC [15], based feedback of the reference to desired state of the system and individual system component. We intend to achieve this by incorporating constituent cost models that capture the dynamical properties of the Infinite Cloud. The scheduler also takes into account the computational complexity and stability in order to find an optimal solution and to avoid frequent unnecessary migration. The holistic goal is to have a fully autonomous distributed management system that proactively manages all of the Infinite Cloud's resources [16]. The autonomous management system of each constituent component of Infinite Cloud decides where to place and how much resource to allocate to the applications based on reevaluation triggered by new application deployment or change in its workload dynamics.

A federated MNO cloud can contain thousands of nodes. As such, all nodes cannot be considered for each placement decision in case of tree structured network scenario. Each search placement for each application is assumed to be limited to a subset of all DCs, that satisfies the applications upstream and downstream *bandwidth*, *latency*, *compute*, and *storage* requirements and is a set in which a global minimum can be attained. Assuming that an application best serves its *end-users* in a DC one or several nodes along one or several edges leading to its *end-users*, or of equal distance. The search domain is thus relaxed to not include every node in the network. The capacity of the DCs is assumed to be diminishing with distance from the trees root and reach a minimum at the edge of the network, closest to the *end-users*.

5 Acknowledgements

This work is funded in part by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. Maria Kihl and William Tärneberg are members of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council (VR) and the Excellence Center Linköping - Lund in Information Technology (ELLIIT).

References

[1] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, pp. 1045–1051, 2010. [Online]. Available: http://comjnl.oxfordjournals.org/content/53/7/1045.abstract

[2] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4:1–4:11, July 2009.

[3] R. Shea, J. Liu, E.-H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *Network, IEEE*, vol. 27, no. 4, pp. 16–21, July 2013.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[5] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[6] P. Bosch, A. Duminuco, F. Pianese, and T. L. Wood, "Telco clouds and virtual telco: Consolidation, convergence, and beyond," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 982–988.

[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568

[8] M. Kihl, E. Elmroth, J. Tordsson, K.-E. Årzen, and A. Robertsson, "The challenge of cloud control," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, 2013.

[9] C. Adam and R. Stadler, "Service middleware for self-managing large-scale systems," *IEEE Transactions on Network and Service Management*, vol. 4, no. 3, pp. 50–64, Dec 2007.

[10] K. J. Åström, P. Albertos, M. Blanke, W. Schaufelberger, and A. Isidori, *Control of complex systems*. Springer-Verlag, 2001.

[11] E. W. Dijkstra, "Self-stabilization in spite of distributed control," in *The 8th International Workshop on Feedback Computing (Feedback Computing '13)*. Springer, 1982, pp. 41–46.

[12] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte wireless virtualization and spectrum management," in *Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP*, Oct 2010, pp. 1–6.

[13] S. Singh and P. Singh, "Key concepts and network architecture for 5g mobile technology," *International Journal of Scientific Research Engineering & Technology (IJSRET)*, vol. 1, no. 5, pp. 165–170, 2012.

[14] A. ElNashar, M. El-saidny, and M. Sherif, *Design, Deployment and Performance of 4G-LTE Networks: A Practical Approach*, 1st ed. Wiley, 5 2014. [Online]. Available: http://amazon.com/o/ASIN/1118683218/

[15] 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 *IEEE International Conference on Services Computing (SCC)*, July 2011, pp. 72–79.

[16] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan 2003.

Paper IV

Dynamic Application Placement in the Mobile Cloud Network

To meet the challenges of consistent performance, low communication latency, and user mobility, we foresee a Mobile Cloud Network that incorporates public cloud infrastructures with compute resource augmented Telecom nodes in the access network. We identify key resource management challenges for this Mobile Cloud Network and develop system models for data centres, networks, cloud applications, and user mobility. Based on these models, we define an application placement optimisation problem that incorporates aspects of network link capacity, desired user latency and user mobility, as well as data centre resource utilisation and server provisioning costs. We propose an application placement and migration algorithm based on local search. The proposed algorithm is evaluated in a simulated Mobile Cloud Network environment for three scenarios, user mobility, diurnal usage patterns on a university campus, and precipitous application popularity. Our evaluation demonstrates that the placement and migration algorithm improves application latency, resource utilisation, data centre costs and as such demonstrates the viability of the dynamic resource management of a Mobile Cloud Network.

William Tärneberg, Amardeep Mehta, Eddie Wadbro, Johan Tordsson, Johan Eker, Maria Kihl, Erik Elmroth
"Dynamic Application Placement in the Mobile Cloud Network,"
Work in progress.

1 Introduction

Multi-component and elastic applications are becoming more seamless, different application components are now executed in geographically distributed DCs in order to meet their individual performance and latency requirements for its geographically dispersed set of users. The location and cost is often brokered in a geographically distributed cloud infrastructure [1]. Moreover, cloud applications are accessed at an increasing rate from the resource constrained MDs [2]. The current geographic separation between the MDs and the DC in which the application is hosted, introduces unwanted communication delay and jitter. Moreover, due to intermittent cloud service availability and heterogeneous latency, an MD resort to executing the majority of an application locally in the MD, as opposed to in the cloud [3]. The resource starved MDs can be augmented with resources through heterogeneous private or public cloud resources and peer devices in the network. The local applications residing inside the MD can be offloaded in the form of remote code execution, or in the form of more immersive cloud applications that behave like smart phone apps. The boundary between discrete application components and how they interact with users and objects in their vicinity is becoming increasingly opaque. Cloud applications such as those with dynamically generated content and latency-critical control process require what is today unattainably low communication latency.

Geographically distributed cloud is in widespread use today, for example, Amazon has DCs in North America, Europe and Asia hosting applications for that specific region. Nevertheless, the DCs are still accessed over the best-effort intermediate Internet. The network capability and degree of ubiquitousness of the distributed cloud can be augmented by Telecom-Networks supplemented with compute nodes owned by the MNOs. We use the term MCN to denote the whole system that incorporates public clouds and an MNO's network. An MNOs typically provides voice, data, and text messaging services as an utility. As a new source of revenue MNOs are seeking to provide computing as an utility [4], as a platform host for 3rd-party applications and to augment capacity of the resident MDs in its network. The supplemented compute resources can also be used for virtualising Telecom-infrastructure and services.

Hosting heterogeneous services with different individual objectives as well as the holistic management objectives of an MNO of its MCN is to ensure that it persistently meets the SLAs of the applications it hosts, while minimising its total resource usage. The highly distributed and heterogeneous nature of an MCN introduces several interesting resource management challenges arising from a highly dynamic workload, heterogeneous energy costs and resources, rapid user mobility, and multi-component applications, [5].

In this paper we study the feasibility of an MCN infrastructure by evaluating application placement algorithms in such infrastructure. With the highly mobile nature of the workload an MCN is subjected to, we focus our attention on where to run appli-

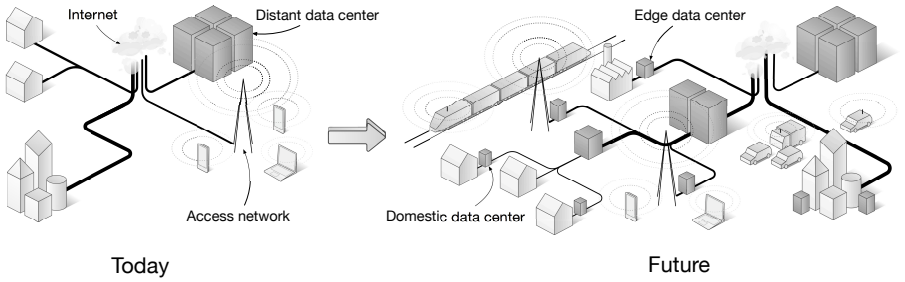


Figure 1: Present versus MCN

cations in the network and how to continuously evaluate that decision as an instrument to fulfill the holistic management objectives of an MNO. To this effect, we propose an objective function to minimise the global system cost to manage the DCs and the network resources of the MCN when hosted in a tree-structured network topology. Furthermore, we present experiments designed to study the validity of the placement algorithms in the MCN paradigm. To do so, we have constructed constituent models to capture the fundamental properties of an MCN.

This paper has the following structure: Section 2 highlights the resource management challenges facing the MCN. Section 3 describes mathematical models associated with the MCN. Section 4 formulates our problem as an optimization problem. Section 5 describes Application placement methods based on the objective function described in the earlier section. Section 6 presents details about the composition of the targeted workload scenarios, experiments to evaluate the placement algorithms and their results. Section 7 covers related work in tangent research fields. Section 8 summarises this paper's contributions, conclusion of the experiments and future work.

2 Resource Management Challenges

In this section, we cover the resource management challenges that an MCN may face. We begin by introducing MCN's service offerings and its resource management objectives.

One of the foremost challenges in the MCN paradigm is how to manage the highly heterogeneous and distributed resources in a complex system. The sheer size of the infrastructure and the number of management parameters renders a fully centralised resource allocation strategy infeasible [6]. As a result, a decentralised collaborative resource management approach needs to be considered. Before we can begin to design a distributed management approach we explore the theoretical optimal solution. A

centralised system will be able to provide an optimal management solution but might be practically infeasible due to, for example, computational complexity, scalability, and fault tolerance. In the hypothetical and experimental scope of this paper we are free from those constraints.

2.1 Service paradigm

The components in an MCN can be viewed as components in a federated cloud [7] whose resources are brokered globally but are for example sought after for their locality to a specific group of users, sensors, and/or actuators. Application components are submitted by application owners from beyond and within the network to serve a subset of the network's population. Application owners impose performance, availability, latency, and locality requirements on an MCN in the form of a SLO or a Service Level Agreement (SLA). Additionally, both end-users and application owners alike are agnostic to where the application is hosted and how the network and cloud infrastructure is managed. The end-users cannot impose requirements on the network or on the applications' performance. Performance requirements, SLOs, for applications originating from an end-user device, are defined by the application owners. Additionally, there is no resource competition between applications and an MCN honours no priorities, but rather, applications have global performance objectives where an MCN might augment performance and facilitate scalability. The decision to submit an application to an MCN is therefore assumed to be made by the application owner.

The MNO's overall management objective is to ensure that the fundamental circumstances for an application to perform according to its SLO or SLA are met. As such, an MCN operator practices admission control and can reject new applications if the application will compromise its internal management objectives and the integrity of the other applications' SLO or SLA. Once an application component has been placed, the performance of an application is the result of the applications' properties and the internal resource management policies of the DC the application is running on, and is beyond the scope of this work. Note that we are thus not concerned with VM or application to PM mapping. Application components hosted by an MCN are assumed not to have a scheduled deadline, but are rather being terminated based on the application's internal management objectives.

2.2 Resource management objectives

The management objectives for an MNO operating an MCN are similar to those found when operating a wireless network, such as user mobility and limited network capacity. Nevertheless, there is a clear paradigm chasm between Telecom and cloud services. Telecom provided services such as voice have very well defined and strict SLAs. Depending on the service type, cloud service SLOs on the other hand are more

loosely defined, where the service offering is multidimensional and given the nature of the resource offering, performance responsibility is more opaque [8]. Additionally, operable core and access networks are prerequisites to hosting and accommodating cloud resources and traditional Telecom services in the network. As such, the successful operation of the access and core networks are therefore prioritised over the cloud services. The scope of this paper does therefore not cover MNO services and network infrastructure virtualisation, as their objectives overlap with that of an MCN.

We make the assumption that an MNO's MCN is managed on top of the existing Telecom infrastructure. The MCN management process is agnostic the momentary load and objectives of the Telecom network. The objective of an MCN management entity is therefore to minimise the resource usage and thus the resulting operational cost and incurred load on the shared Telecom network, and to provide a service with a finite set of resources.

2.3 Challenges

The internal MCN management challenges for an MNO are found in the union of cloud and mobile infrastructure. An MNO has only a few degrees freedom to control the operations of the infrastructure. The MNO can alter:

- The number of applications in the network.
- The pallet of applications and application's heterogeneity.
- Which pieces of infrastructure to run.
- Where to run the application components.

Continuous evaluation of application component placement is the common denominator. We do not suppose to assume how specific applications or set of users behave. We thus assume that any application can behave in any manner in the realm of what is physical and computationally possible. When re-evaluating an application placement, the management process determines if the energy, compute, network, and latency costs fall short of any of the possible placement possibilities that qualify, for a certain period of time. The systems' rate of change determine the duration under which a decision is valid. The number of possible placement combinations and the rapid rate of change means discrete placement decisions need to rely on the prevailing workload, resource availability, and user location. The system needs to re-evaluate the placement of application components, whenever workload changes for an application, when new applications arrive or are terminated, when applications scale up or down, or when foreground traffic volumes change. The triggers and decisions need to be at the granularity of individual application components.

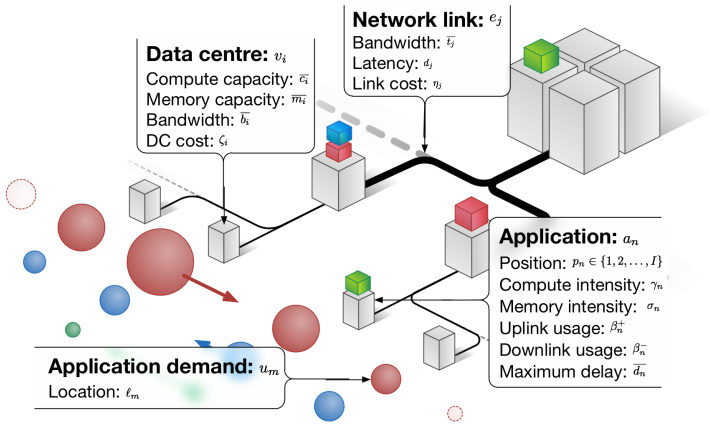


Figure 2: Model overview

Operating a profitable network relies on an operational network and the ability to cost-manage that network. The resource heterogeneity of an MCN infrastructure, the mobility of the users in the network introduce and the heterogeneity of the applications to which they subscribe introduce a complex set of management decisions. The MNO will need to manage the placement decision and the subsequent placement revaluation of application components in a manner that minimises the overall resource usage.

3 System model

In this section we propose a system model of an MCN for experimentation. We begin by constructing a general model for an abstracted MCN and its topology, followed by a more detailed model for each component in the system.

The topology depicted in Figure 1 reflects the union of an MNO's network and a federated cloud infrastructure and should be seen as an abstraction of an MCN topology proposed in [4]. The placement and scale of an MCN's' DCs will be heavily dependent on the degree of an MNO's infrastructure virtualisation [9], the degree of convergence of core and access networks, and the prevailing geographic demand for proximal compute capacity. Although some bounds can be constructed, these properties are not yet defined as the design of forthcoming mobile access network standards and topologies are far from being finalised [10]. Additionally, we make no assumptions of placement or scale but our models are generic enough to handle a number of feasible next-generation infrastructure topologies.

As a whole, we model an MCN as an undirected forest or tree graph [11, 4, 12, 13], where the vertices are DCs and the edges are network links, each with a set of finite

resources, see Figure 2. Applications are hosted in a DCs and are subject to demand through the network links, originating at the leaf nodes. The graph $G = (V, E)$ denotes a tree depicting the MCNs network topology, where

$$\begin{aligned} V &= \{v_i \mid i = 1, 2, ..., I\}, \\ E &= \{e_j \mid j = 1, 2, ..., J\}, \end{aligned} \tag{1}$$

where v_1 is the root node. The subscript i of the node v_i is named such that all the nodes v_k between v_i and v_1 follows

$$(v_k, v_1) \leq (v_i, v_1), \ \ \forall k \leq i, \tag{2}$$

where the distance between nodes $v \in V$ and $w \in V$, (v, w) is measured in number of vertices that is on the shortest path from v to w. The leaf vertices are connected by RBSs from which the the end-users access the network. Thus, the leaf vertices are geographic aggregation points of application demand.

3.1 Data centre Model

The MCN compute resources will proposedly reside in existing MNO infrastructure [4], such as in an MNO's regional offices or what remains of previous generation network infrastructure. Furthermore, the compute capacity is proportional to the aggregate demand of applications from their users that have access to it, thus successively decreasing with depth. Henceforth, compute cost will increase with depth.

Each vertex is a DC and hosts applications using a set of finite resources. Vertex v_i, $i \in \{1, 2, \ldots, I\}$ in the graph has the following features:

- **Compute capacity** $\overline{c_i}$, a number describing the total compute capacity of the DC.
- **Memory capacity** $\overline{m_i}$, a number describing the amount of memory on the DC.
- **Bandwidth**, $\overline{b_i}$, a number describing the maximum throughput that the DC can handle.

In addition to the features above, vertex v_i, $i \in \{1, 2, \ldots, I\}$ is associated with the following operational cost

- **DC cost** ζ_i, a function of resource usage (compute, memory, and bandwidth) that returns the DC's running cost per time unit.

In general, the leaf vertices of the graph correspond to smaller DCs and thus the compute costs are arguably greater at the leaf vertices than at vertices at lower depths in the tree, [14].

3.2 Network Model

Existing 3rd and 4th generation mobile access networks are generally tree-structured [15]. This structure will feasibly be inherited by future mobile infrastructure generations. Furthermore, bandwidth availability as well as communication latency and jitter decrease with tree depth. Additionally, the network topology is modelled as a tree structured graph, where each edge has network resources and exhibits latency and congestion.

Each edge e_j, $j \in \{1,2,...,J\}$, in the graph has the following features

- **Bandwidth** $\overline{t_j}$, a number specifying the maximum throughput over the edge.
- **Latency** d_j, a function of the throughput that returns the delay caused by that link's resource utilisation and length.

In addition, each edge has the following operational cost

- **Link cost** η_j, a function of throughput that returns the link's running cost per time unit.

The communication latency of an application is dictated and maintained by the applications' relative locations to its demand and the level of congestion on the links it employs. As the demand mobility from one edge node to another can be highly dynamic, the size and location of applications' demand can vary with time. Latency is a modelled as a function of propagation delay and network congestion [16]. In general, the bandwidth cost increases with the distance to the root vertex.

3.3 Application Model

An MCN hosts applications a_n, where $n = 1,2,\ldots,N$. We let $A = \{a_n \mid n = 1,\ldots,N\}$ denote the set of all applications hosted by an MCN. Application a_n, $n \in \{1,2,\ldots,N\}$, see Figure 3, has the following features:

- **Position** $p_n \in \{1,2,\ldots,I\}$, a number specifying that the application component is running on the DC at vertex v_{p_n}.
- **Compute intensity** γ_n, an increasing function of the demand of the application component that describes the amount of computational resources required by the application component.
- **Memory intensity** σ_n, an increasing function of the demand of the application component that returns the amount of memory required by the application component.

- **Uplink usage** β_n^+, an increasing function of the demand of the application component as well as the locations of the application component's end-users that returns the uplink throughput associated with the application component,
- **Downlink usage** β_n^-, an increasing function of the demand of the application component as well as the location of the application component's end users that returns the downlink throughput associated with the application,
- **Maximum delay** $\overline{d_n}$, the longest delay (latency) that provides the users of the application component a satisfactory experience. This delay is specified in the SLA between MNO and application owner.

Applications thus scale vertically in a DC, as a function of demand.

We remark for clarity that in the experiment section of the current work, we only consider single tier applications. However, the model can be generalised to account for multi-tier applications. In the multi-tier setting, application a_n can be considered as being composed of s_n stages or sub-applications. Each of these sub-applications will have the same features as a single tier application. The relationship between application components is expressed with a demand affinity as described in [17]. As an alternative to the viewpoint of handling multi tier application would be to include general affinity and anti-affinity constraints between application components.

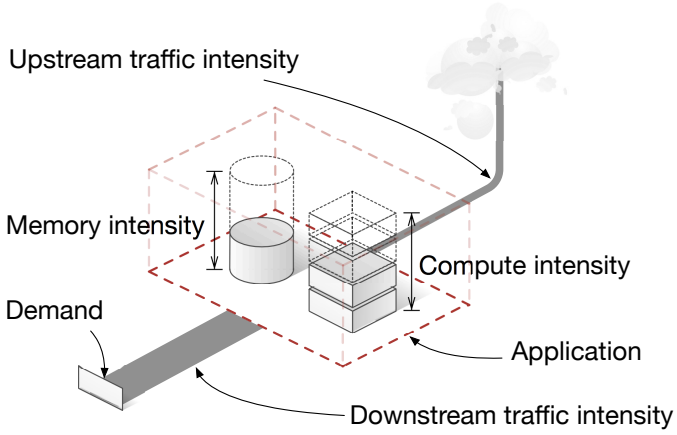


Figure 3: Application model

3.4 User model

Finally, we let $U = \{u_m \mid m = 1,2,\ldots,M\}$ be the set of users of an MCN. Each user u_m, $m \in \{1,2,\ldots,M\}$ has the following features

- **Location** $\ell_m \in \{1,2,\ldots,I\}$, a number specifying that the user is currently served by the DC at vertex v_{l_m}.
- **Active applications** $A_m \subset A$, a set of application components that the user currently runs.

For future notation, we define $U_n = \{u_m \in U \mid n \in A_m$ to be the demand for an application component n and let $U_{n,i} = \{u_m \in U_n \mid \ell_m = i\}$. Note that for the numerical experiments in this paper, we do not explicitly track the users but only the demand of each application component at each leaf vertex.

4 Optimisation Formulation

In this section we detail our proposed MCN application placement principals. The infrastructure is restricted by resources with finite capacity and can thus not host an infinite number of applications. Furthermore, not all placement constellations can meet all application constrains. Accommodating an application's constraints and meeting its SLA/SLO is a prerequisite to generate revenue. Therefore, analogously, when we search for possible placement options we first impose the application's constraints on the set of available DCs and the subsequent intermediate links. Applying the application's resource and performance constrains prunes the search tree. Thereafter, we search for the set of placement options that incurs the least overload in that point in time.

4.1 Resource Metrics and Constraints

The main aim of the optimisation in this paper is to decrease the running costs of an MCN by placing/moving the applications on different DCs by normalizing the usage of the infrastructure's DC resources and minimizing the incurred network usage. The control or decision variable here will be the vector $\mathbf{p} = (p_1, p_2, \ldots, p_N)$ that holds the position of all applications. For full generality, we let the set of admissible placements be

$$\mathcal{A} = \left\{\mathbf{p} \in \mathbb{Z}^N \mid p_n \in \{1,\ldots,I\}\right\}. \tag{3}$$

Below, we describe how some important resource metrics defined at the vertices (DCs) and edges (links) can be computed from the features of the four models introduced in the previous section. To detail the relations, we need to define the following

objects: Let $P_{i,i'}$ denote the path from DC i to DC i', that is, the set of edges that connects the two DCs. Moreover, let $E_i = \{e \in E \mid v_i \in e\}$ be the set of edges that represents links that are connected to DC i.

The throughput t_j over link j is the total usage (uplink plus downlink) for all pairs of users and applications such that the link is on the path connecting the DC serving the user and the DC hosting the application. More precisely, the throughput is given by

$$t_j = \sum_{\left\{n,m \,\middle|\, \substack{u_m \in A_n, \\ e_j \in P_{p_n,\ell_m}}\right\}} \left(\beta_n^+\left(|U_{n,\ell_m}|\right) + \beta_n^-\left(|U_{n,\ell_m}|\right)\right). \tag{4}$$

For each application $a_n \in A$ and user $u_m \in U_n$ of that application, the latency $d_{n,m}$ experienced by the user is the sum of all latencies on the path connecting the DC serving the user and the DC hosting the application. Thus, the latency $d_{n,m}$ can be computed as

$$d_{n,m} = \sum_{\{j \mid e_j \in P_{p_n,l_m}\}} d_j(t_j). \tag{5}$$

We do not include extra latency incurred due to intermediate DCs. The compute and memory usage at DC i is the sum of the corresponding usage by the applications that are running at the DC, that is,

$$c_i = \sum_{\{n \mid p_n = i\}} \gamma_n(|U_n|) \tag{6}$$

and

$$m_i = \sum_{\{n \mid p_n = i\}} \sigma_n(|U_n|). \tag{7}$$

The throughput over vertex i is the sum of the throughputs of all edges that are connected to the corresponding DC. Thus,

$$b_i = \sum_{\{j \mid e_j \in E_i\}} t_j. \tag{8}$$

Each of the resource metrics is associated with a constraint connected to the features of the DC, application, and network models. These constraints are

$$c_i/\overline{c_i} \leq 1, \qquad i = 1,2,\ldots,I, \tag{9}$$

$$m_i/\overline{m_i} \leq 1, \qquad i = 1,2,\ldots,I, \tag{10}$$

$$b_i/\overline{b_i} \leq 1, \qquad i = 1,2,\ldots,I, \tag{11}$$

$$t_j/\overline{t_j} \leq 1, \qquad j = 1,2,\ldots,J, \tag{12}$$

$$d_{n,m}/\overline{d_n} \leq 1, \qquad n = 1,2,\ldots,N \text{ and } \{m \mid u_m \in U_n\}. \tag{13}$$

All constraints above are written on the form $a/\overline{a} \leq 1$, that is that the relative usage (or latency) of a particular resource should be at most 1. The choice of a common form simplifies the discussion about the formulation of the optimisation problem below.

4.2 Optimisation problem

We design the objective function in Equation (14) in order to capture the application execution cost and the overload penalty on the node as well as edge resources in the system. Here, the objective function is constructed from the infrastructure providers' viewpoint. Primarily, they want to minimize the overall running cost.

$$J(\mathbf{p}) = \sum_{i=1}^{I} \zeta_i(c_i, m_i, b_i) + \sum_{j=1}^{J} \eta_j(t_j). \tag{14}$$

In this work, we assume that the cost for running the DCs is linearly proportional to their compute resource usage and that the cost for the network is linear to the throughput over each link. Remark that adding a constant background cost that represent the cost for when the links and DCs are idle does not influence the possible savings by migrating the applications. Thus, in principle, we formulate our optimisation problem as

$$\min_{\mathbf{p} \in \mathcal{A}} J(\mathbf{p}), \tag{15}$$

subject to constraints (9–13).

The problem formulation above is straightforward and intuitive. However, we are dealing with a system where the usage of the application will vary with time and thus a small change in the demand may yield a previously optimal solution infeasible. In the worst case, major migrations would be required to resolve the infeasibility. To ensure feasibility of the constraints stating that the relative usage should be smaller than 1 and to avoid link or vertices becoming overloaded, we introduce a penalty initialization point $\tilde{x} < 1$ and define the penalty–barrier function

$$f_{\tilde{x}}(x) = \begin{cases} 0, & \text{if } x < \tilde{x}, \\ \frac{1}{1-x} + \frac{2\tilde{x}-x-1}{(1-\tilde{x})^2}. & \text{if } x \geq \tilde{x}. \end{cases} \tag{16}$$

In the equation above, x should be viewed as the relative usage (or latency) that is the quotient on the left hand side in constraints (9–13). For the case when $x \geq \tilde{x}$, the first term is selected to ensure that $f_{\tilde{x}}(x) \to \infty$ as $x \to \infty$ and the second term is select so that $f_{\tilde{x}}(\tilde{x}) = f_{\tilde{x}}(\tilde{x}) = 0$, that is, to guarantee that $f_{\tilde{x}}$ is continuously differentiable in the interval [0,1).

By construction, the function $f_{\tilde{x}}$ acts as both a penalty and a barrier function; it is a penalty function for constraints of the type $x \leq \tilde{x}$ and a barrier function for the constraint $x \leq 1$. In essence, this makes it easy to modify the point where the penalisation

starts for each constraint separately. Here, we utilize this versatility by having different penalty initialisation points for constraints corresponding to different features. A more elaborate set-up could for example have a penalty initialisation point for the compute resource usage that depends on the level in the tree that the corresponding DC is located. To compute the overall penalty G, we add the penalty–barrier function corresponding to all constraints with their respective penalty initialisation points. That is,

$$\begin{aligned} G(\mathbf{p}) = &\sum_{i=1}^{I} f_{\tilde{c}}\big(c_i/\overline{c_i}\big) + \sum_{i=1}^{I} f_{\tilde{m}}\big(m_i/\overline{m_i}\big) + \\ &+ \sum_{i=1}^{I} f_{\tilde{b}}\big(b_i/\overline{b_i}\big) + \sum_{j=1}^{J} f_{\tilde{t}}\big(t_j/\overline{t_j}\big) + \\ &+ \sum_{n=1}^{N} \sum_{\{m|m\in U_n\}} f_{\tilde{d}}\big(d_{n,m}/\overline{d_n}\big), \end{aligned} \tag{17}$$

where $\tilde{c}$, $\tilde{m}$, $\tilde{b}$, $\tilde{t}$, and $\tilde{d}$ are the penalty initialisation points corresponding to the constraints for compute usage, memory usage, DC throughput, link throughput, and application latency, respectively. Here, we have simply added the individual penalty–barrier functions for all constraints. This corresponds to taking the 1-norm of the vector whose elements hold the values of the individual penalty–barrier functions for all constraints. An alternative overall penalty–barrier method is obtained by taking another vector norm of the constraint vector.

To conclude, rather than solving problem (15) subject to constraints (9–13), we solve the following problem

$$\min_{\mathbf{p}\in\mathcal{A}} J(\mathbf{p}) + \mu G(\mathbf{p}), \tag{18}$$

where μ is a positive penalty parameter and J and G are defined in expressions (14) and (17). Dimension for J is in terms of monetary cost per time unit, whereas G is dimensionless. Hence, the unit for μ is monetary cost per time unit.

5 Application Placement Methods

One approach to solve the optimisation problem (18) is to perform a so-called exhaustive search, that is to evaluate the objective function for each admissible placement. The computational complexity of this approach is exponential, since $|\mathcal{A}| = N^I$. For example, to evaluate the objective function in a setting with 22 applications and 12 DCs would require $O(10^{18})$ summation operations (each evaluation of the objective function requires $O(10^2)$ summation operations). Thus, even for relatively small problems this approach is computationally intractable. Here, we use a local search algorithm, described below, to find approximate solutions to optimisation problem (18).

We re-evaluate the objective function, $J + \mu G$, during run time in order to compute the total cost for running all the applications inside the infrastructure. The re-evaluations are triggered either by an internal event in the infrastructure, entity, or application or is triggered by a periodic heartbeat signal, agnostic to the state of the system.

For each $\mathbf{p} \in \mathcal{A}$, we define the k-neighbourhood of $\mathbf{p}$ as

$$\mathcal{N}_{\boldsymbol{p}}^{k} = \{\boldsymbol{q} \in \mathcal{A} \mid \|\boldsymbol{p} - \boldsymbol{q}\|_{\mathcal{A}} \leq k\}, \tag{19}$$

where $\|\cdot\|_{\mathcal{A}}$ is a measure of the network distance for elements on $\mathcal{A}$. A larger k gives more freedom to migrate, replicate, and consolidate application components; however, we need to select k such that $|\mathcal{N}_{\mathbf{p}}^{k}|$ is not too large. The network distance can be computed as,

$$||p - q||_{\mathcal{A}} = \sum_{n=1}^{N} \delta_{\mathcal{A}}(p_n, q_n), \tag{20}$$

where a possible choice of function $\delta_{\mathcal{A}}$ is:

$$\delta_{\mathcal{A}}(p_n, q_n) = \begin{cases} 1, \text{ if } p_n \neq q_n, \\ 0, \text{ else .} \end{cases} \tag{21}$$

An alternative choice is:

$$\delta_{\mathcal{A}}(p_n, q_n) = \begin{cases} |P_{p_n,q_n}|, \text{ if } p_n \neq q_n, \\ 0, \text{ else ,} \end{cases} \tag{22}$$

where $|P_{p_n,q_n}|$ can be computed considering the latency in all the traversed edges.

We employ a depth first search algorithm to find the neighbourhood nodes in the tree. We start with finding all the solutions with neighbourhood depth k and compute the local optimal solution. From the local solution we construct its subsequent $k-$neighbourhood solutions and compute the local optimal solution. We repeat the process until we achieve the specified maximum number of iterations or the cost starts increasing. The algorithm is described as,

For larger neighbourhoods, that is when k is large, branch and bound techniques can be used to efficiently solve the local optimisation problem in Line 6 in Algorithm 1.

Re-evaluation interval

At each time stamp of evaluation, all applications are simultaneously evaluated to find the constellation with the lowest global overload and cost. A local or exhaustive search is performed over the possible placement options to find the placement. No discrete

Algorithm 1 Local Optimisation or Local Search

Input : current placement vector of applications (**p**), maximum number of iterations (**maxIter**), depth (**k**)
Output : new placement vector for applications
$C_{\mathbf{p}} \leftarrow J(\mathbf{p}) + \mu G(\mathbf{p})$
nIter $\leftarrow 0$
while nIter $<$ **maxIter do**
    $\mathcal{N} \leftarrow$ get $\mathcal{N}_{\mathbf{p}}^{k}$, the k-neighbourhood of **p**
    $C_{\mathbf{q}} \leftarrow \min_{\mathbf{q} \in \mathcal{N}} J(\mathbf{q}) + \mu G(\mathbf{q})$
    if $C_{\mathbf{p}} \leq C_{\mathbf{q}}$ **then**
        break
    end if
    $C_{\mathbf{p}} \leftarrow C_{\mathbf{q}}$
    $\mathbf{p} \leftarrow \mathbf{q}$
    nIter $\leftarrow$ **nIter** $+ 1$
end while
return p

events are identified, such as spikes or spatial shifts in demand. Nor does it make any predictions on the workload or resource availability. The system only acts on the momentarily incurred overhead.

6 Evaluation

In this section, we detail the simulation set-up, some relevant example scenarios and their workload generation. We then proceed with presenting and discussing some results. We begin by describing the different dimensions of the experiments, such as the simulator parameters, workload, and search depths, evaluation frequencies and domains of the placement algorithms. The experiments are carried out using a well-known, coarse-grained simulator designed to capture the fundamental dynamics of an MCN with the workloads specified in Section 6.2. When application components arrive at the system they need to be placed in one of the system's DCs so that clients can access them. In Section 6.2, we first describe an application demand model based on different scenarios, and we then define types of application based on their resource consumption, e.g. CPU intensive or IO intensive. In all the below placement policies, this initial placement is performed by searching for the globally optimal location for all applications which minimises the objective function detailed in Section 4. Finally, the Results section evaluates how attainable a non-divergent feasible system is by analysing how well the different placement algorithms constrain cost and resource

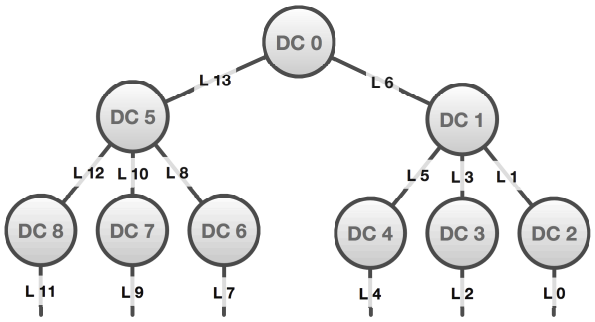


Figure 4: Experiment infrastructure topology

usage with a transient workload.

6.1 Hypothesis and evaluation

To form system performance boundaries, we evaluate the placement of each application at run-time in one of a number of different ways, both optimally and sub-optimally. We begin with statically placing each application, as they arrive, optimally in the DC which minimises the global initial cost, irregardless of the forthcoming workload. This will reveal how much the cost diverges and will quantify the need for periodic re-evaluation of the placement of all applications. Secondly, we regularly re-evaluate the location of all application to minimise total system cost. The latter will provide an optimal lower bound. Conversely, placing applications at random in the infrastructure will provide a divergent upper bound.

Apart from the objective function, the placement algorithms have two degrees of freedom. We alter the search depth for each application from where it is currently placed, limiting its migrations options in the tree. We define the domain and the set of nodes which is evaluated for each application as its neighbourhood. For the experiment, a topology of depth 4, see Figure 4, produces neighbourhood nodes as a set of nodes that include all DCs in the network for an application.

Similarly, alternating the rate of re-evaluation adds a second degree of freedom. In our experiments we alter the re-valuation rate from static to the rate at which the workload is changing. Re-evaluating at higher rate may lead to non-optimal solution.

To evaluate the performance of the placement methods, in our experiments, we observe a number of performance metrics including; the overall heuristic system cost, resource utilisation, and application RTT. The overall cost will tell us how far a method is from the optimal solution. The resource utilisation will tell us how well the method is meeting the system's objective of minimising the incurred network usage. Lastly, the measured application RTT will reveal to what extent the application's performance

will be penalised by a specific placement method.

6.2 Simulation for Application Demand

We construct the application demand model using four parameters that enables us to capture each scenario: time variation, spatial variation, popularity distribution among applications, and resource usage diversity among applications.

To examine the system's behaviour we employ three example scenarios that we consider relevant to an MCN. We begin with a scenario with a high degree of *User Mobility*, where demand for each application is concentrated to a few leaf nodes and is highly mobile. Second, we define a diurnal-centric *University Campus* scenario, where demand of all applications is concentrated at the University during day time and disperses geographically to a uniform distribution at night. Furthermore, our third scenario captures a *Sporting Event* where an application becomes popular in a small group of nodes due to a very high local demand.

In the *User Mobility* scenario, we model the spatial demand distribution as normal over few consecutive nodes to show demand from a batch of users for an application.Time-variation is modelled as a random walk, traversing the consecutive nodes next to the nodes where demand is currently residing. We do not vary the amount of demand, but the spatial distribution changes with time among the nodes having the demand for the application. Furthermore, we model the popularity distribution among applications as uniform. This workload captures the behaviour of a small number of groups of users subscribing to one application each, on consecutive nodes and their independent movement in their locality. This situation resembles that of hyper-local demand emanating to a locally popular application, such as Augmented Reality and autonomous vehicle applications.

In the *University Campus* scenario, we model the demand for an application across the network's leaf nodes as normal. To capture the students' pronounced diurnal migration patterns [18], we vary the standard deviation (σ) diurnally to achieve a nighttime near-uniform distribution with higher σ to a noon distinctly normal distribution with lower σ.The demand for all applications is higher at the nodes near the university campus during day time. We model the popularity among applications as Zipf distribution [19, 20]. The demand variation from uniform during night to normal during day is modelled as linear.

During a *Sporting Event*, demand on the nodes corresponding to the places the where the teams are based or where they are playing is likely to increase as fans and spectators start browsing for scores and/or stream the game to their MDs.

We model the unexpected surge in demand of these applications as a spike. In our previous work [21], we analysed spikes and defined a spike as a significant shift in the workload pattern. Here, we model the time variation for the growing phase of a spike as,

$$y_\tau = y_{\tau-1}(1+\alpha_g(1-\lambda_g y_{\tau-1})), \tag{23}$$

where y_τ is a time series of demand for an application, α_g and λ_g are coefficients, α_g denotes the growth rate and slope in the log scale. Inverse of λ is the maximum of the popularity. We model the decreasing phase of an spike as,

$$y_\tau = y_{\tau-1}(1-\alpha_d(\lambda_d y_{\tau-1}-1)), \tag{24}$$

where α_d and λ_d are coefficients. Using the model defined above, we construct a spike similar to one experienced during Fifa 1998 world championship, using the parameters, α and λ for each stage of the spike. There are two stages for the growing phase and one stage for the decreasing phase for the first peak, one stage each for the growing and decreasing phase for the second peak. The spike is only present in one node, and is spatially stationary.

6.3 Application types

We model an application and its demand based on its performance goals. *Interactive or transactional applications'* performance goals are based on average or percentile response time and throughput over a short time interval. These applications are managed by flow control, load balancing, and application placement. *Non-interactive or batch applications'* performance goals are concerned with the completion time of the individual jobs that usually require an extended period of time. These applications are managed by scheduling and resource control.

Applications consume heterogeneous resources. Based on the relatively higher specific resource requirement, applications can be CPU, I/O, or memory intensive [22, 23, 24, 25, 26]. An example of a CPU intensive benchmark application is Sysbench, similarly PostMark is a benchmark for IO intensive applications. Furthermore, [27] reports on profiling of applications for VM placement purpose. They used Apache web application, Pgbench and X264 video encoding programme.

6.4 Simulator

We employ an event driven simulator to validate the validity of the topology and the proposed placement methods. Existing simulation frameworks such as NS-3 [28] and CloudSim [29] offer competent network and data centre models, respectively. Nevertheless, neither framework offer a complete solution at the desired abstraction level nor do they scale adequately for large networks. As a result we developed an coarse-grained event-driven simulator in Python that utilises PySim as the event-driven framework. The simulator employs the model detailed in Section 3 which represents an MCN topology with DCs, a network, and time-variant demand. The input

to the simulator is a time series workload, which also progresses time. The workload is propagated throughout the network to the DCs in which the application is hosted, incurring a resource usage on the affected resources proportional to the demand and in accordance with the properties of the application.

The simulator features a centralised management unit that places new applications as they arrive and updates the network with resources consumed by the application. One or multiple parallel controller modules can trigger placement re-evaluations, either periodically or at an arbitrary event. The placement algorithms are those specified in Section 4. The simulator monitors and outputs the momentary total cost, application RTT, and resource utilisation levels.

6.5 Experimental Set-up

In this section we introduce and motivate the parameters that define our experiments. We begin with detailing the scope of the experiments and reason about the size of the simulation. This is followed by a value parametrisation of the models and cost functions described Section 3 and Section 4.

We have simulated a small scale infrastructure consisting of 9 DCs and 6 applications, resulting in $6^9 = 10077696$ permutations per re-evaluation, in the optimal case. The topology has one root DC, 2 intermediate DCs and 6 edge DCs, see Figure 4.

We have simulated the applications' workload demand as described in Section 6.2. Based on the demand, we allocated resources to the nodes and links in the system, so that no application is denied admission to the system due to lack of resources. For each scenario, we computed the total amount of resources required for all the applications at each leaf node at each time instance. We allocated the maximum of the demand of the time-series data over all the nodes to each node so that all the applications have the possibility to run on the leaf nodes. For the nodes other than leaf ones, we allocated resources equal to sum of resources of their children nodes.

In each experiment we evaluate the following placement methods:

- **Random static**, where each application application components is initially placed at random in the network and are not dynamically moved.
- **Random continuous**, where each application at each heartbeat is relocated to a random node in the network.
- **Local continuous**, local minimal cost search with 4 iterations.
- **Global static**, applications are initially placed in the globally optimal node but are not continuously re-evaluated and relocated. The search depth is set to 4.
- **Global continuous**, where each application's placement is continuously re-evaluated and are relocated accordingly. The search depth is set to 4.

6.6 Parameters for the Optimisation function

The optimisation function in optimisation problem (18) contains a weight μ that combine the overall running cost and the overall penalty (or overload cost). The application execution cost for each application on a node is proportional to the resource consumed by the application on the node as well as the operational cost of the node. In the experiments presented in this paper, the operational cost per resource on a large DC is 1.33 times lower than the corresponding cost on the smallest DC, whereas the operational cost per resource on a medium DC is 1.2 times lower than that of the smallest DC.

Let x be the resources consumed by an application. The operational cost will be proportional to $\frac{x}{1.33}$ for a large DC, whereas it will be proportional to $\frac{x}{1.2}$ and x for a medium and small DCs respectively. For the optimization experiments, we use $\mu = 1$ and set the penalty initialization point, $\tilde{x}$ in Equation (16), to 0.7 for node overload and to 0.0 for link overload. Figure 5 illustrates the execution cost and overload cost for our reference simulation system. In the reference system, the capacity ratio between small and medium DCs is 1:3, while the ratio small and large DCs is 1:6; the ratio for the execution cost, for the complete DC, is 1:2.5 between small and medium DCs and 1:4.5 between small and large DCs. The meeting point of the curves determines the dynamics for the cost for running applications in the system.

6.7 Results

In the section below we present our evaluations of the placement algorithms using the workload presented in Section 6.2.

Cost The resulting total system cost reveals how well the placement algorithms are able to minimise their objectives. Figures 6 to 8 reveal the time variant cost for the different placement methods on the different workloads.

For the *Mobile users* workload case, the globally optimal method achieves an average 25% lower cost than when applications are global optimally statically placed, see Figure 6.

Due to the high level of mobility in the *Mobile users* workload we introduce a random placement scheme as a reference. Knowing only that demand is not stationary, one of the simplest strategy we can employ is to randomly place applications in the network. In *Mobile users* workload is transient, the volume and constellation of demand will thus never return to the same quantity and location, respectively. As illustrated by Figure 6, on average, when applications are randomly statically placed, the cost is 5% higher than in the optimal static placement scheme, but comes at a much lower computational cost. The results for the random placement scheme are not

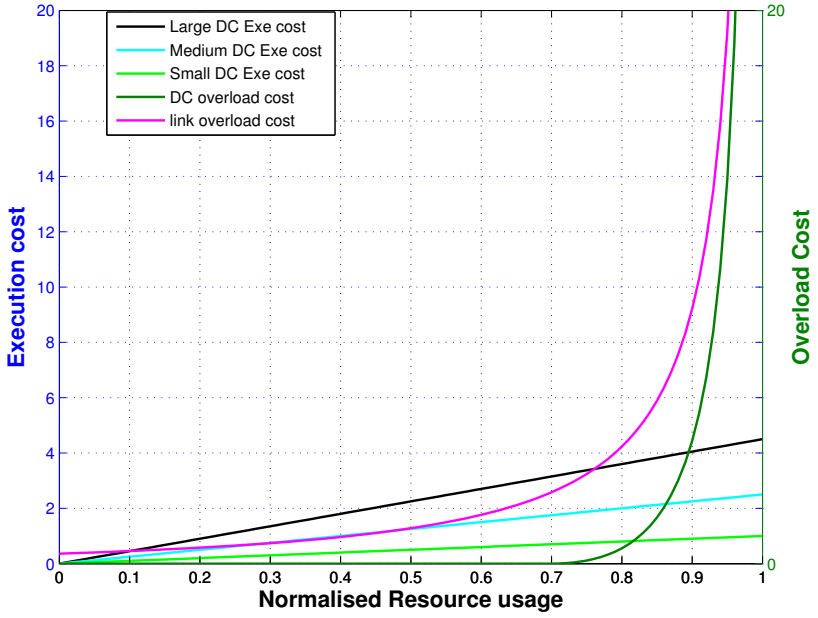


Figure 5: Overload function vs. Resource Execution cost of large and small DC and link

included in Figures 7 and 8 as it produces cost far higher than the globally and locally optimal solutions, which are the primary focus of this paper.

As illustrated by Figure 6, the locally optimal algorithm, with a maximum of 4 iterations, performs near-optimal on the *Mobile users* workload. The cost difference is on average 1%. The discrepancy is due to the temporal and spatial diversity of the workload and the resulting multiple minima. This is confirmed by Figures 7 and 8 where the workloads are less spatially and temporally complex. When applied to the *Campus* and *Sporting event* workload, the cost difference to the globally optimal is on average $< 0.5\%$.

Furthermore, our experiments also show that the search depth has little impact on the system's ability to minimise costs. A depth less than 4 introduces a cost lag compared to optimal but is not divergent.

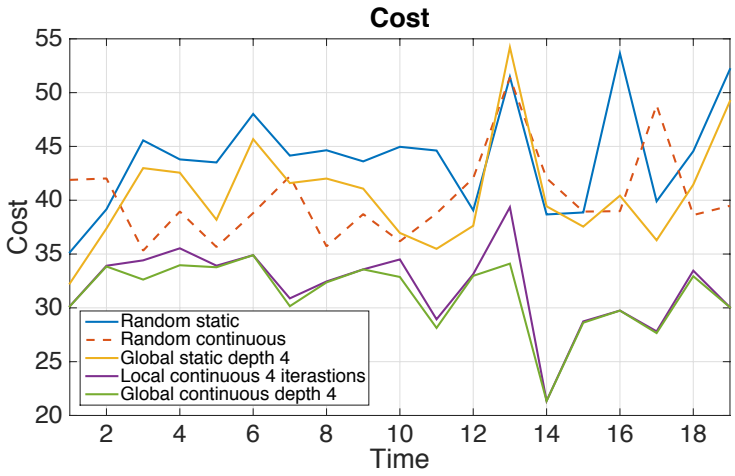


Figure 6: Cost time-series for all placement methods for the *Mobile users* workload

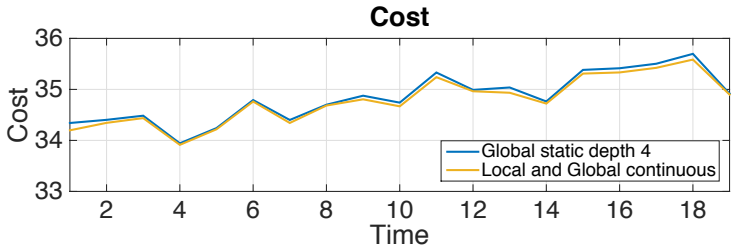


Figure 7: Cost time-series for all placement methods for the *Campus* workload

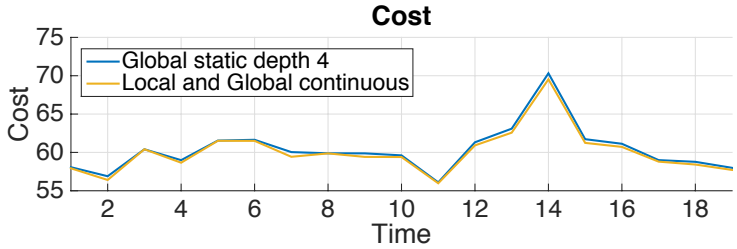


Figure 8: Cost time-series for all placement methods for the *Sporting event* workload

RTT Although none of the applications in this scenario have SLA constraints dictating RTT, the holistic objective to minimise the network utilisation works in favour

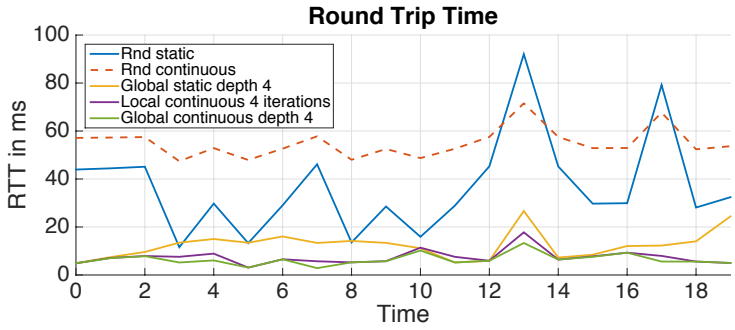


Figure 9: RTT time-series for all placement methods for the *Mobile users* workload

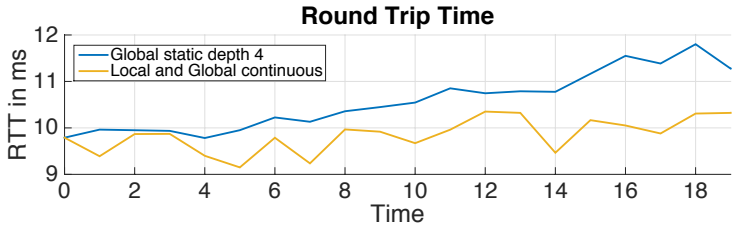


Figure 10: RTT time-series for all placement methods for the *Campus* workload

of each individual applications RTT, as network paths and congestion are minimised.

In the *Mobile users* case, Figure 9 reveals that applications on average experience a 35% lower RTT when applications' placements are continuously evaluated than the case when they are statically placed. Random placement delivers a significantly higher RTT for each application, for the same workload. In the *Campus* workload case, RTT increases 10 fold across the scenario, see Figure 10. The uniformity of demand across all applications and the stationary of the centre of demand leaves little room to significantly alter the placement of the applications, resulting in a gradual decline in RTT as demand concentrates around a few nodes. Running the *Sporting event* workload reveals how the affected application is able to relocate to accommodate the spike and mitigating some of the incurred latency, see Figure 11.

Utilisation As stated in Section 2 the primary objective of any application placement algorithm is to minimise network usage, and to mitigate skewed resource usage of the MNOs' DCs, while meeting the resident applications' SLAs. Figures 12a to 12c reveal that network utilisation is indeed reduced when applications' placements are continuously evaluated. Furthermore, Figure 12a shows that the mean link resource utilisation in the Mobile user case is reduced by 5%. On the other hand, mean DC utilisation

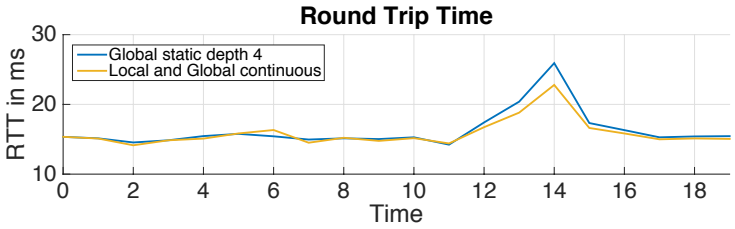


Figure 11: RTT time-series for all placement methods for the *Sporting event* workload

is increased by 5%. In the same workload, the globally optimal algorithm achieves lower mean utilisation levels than the locally optimal algorithm. Furthermore, random static, random continuous, and globally optimal static placement algorithms produce the same utilisation levels within a range of 2%, with a slight advantage to the globally optimal static placement algorithm.

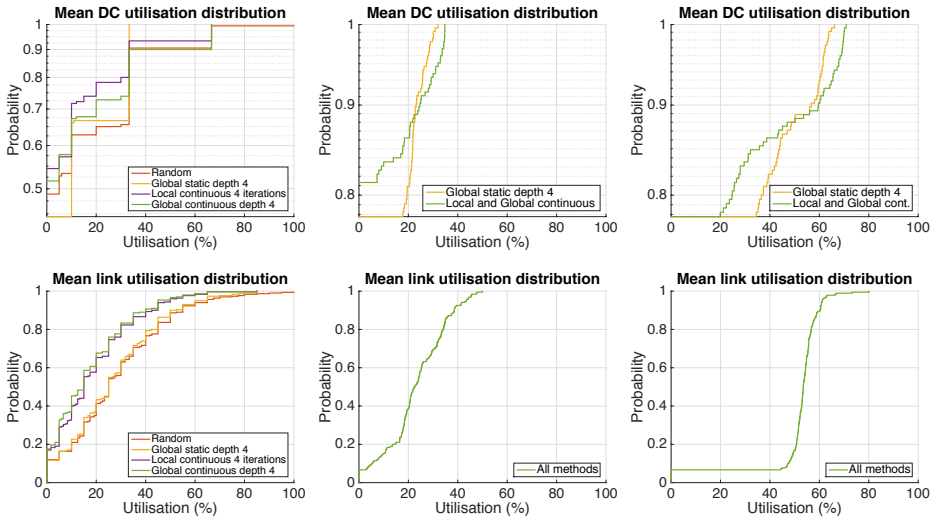
Figures 12b and 12c reveals that the globally and locally optimal algorithms offers no mean link utilisation improvement over the globally optimal static scheme, on the *Campus* and *Sporting event* workloads. Mean DC utilisation is improved using the globally and locally optimal algorithms over the globally optimal static placement scheme, see Figures 12b and 12c. In either case the utilisation levels are kept at a desirable level.

6.8 Complexity Discussion

We carried out a small set of experiments by using our simulator to show the feasibility of an MCN infrastructure. In particular, these experiments illustrate how to do placement of the applications running inside the infrastructure using local optimization algorithms to minimise our objective function. Exhaustive search gives optimal solution but it becomes intractable for s large number of nodes and links. On the other hand, a local search solution gives local optimal solution, but does not guarantee global optimality. We remark that for both the global and the local optimisation algorithms, one can use branch and bound or other heuristics to reduce the search space.

7 Related work

In this section we cover some related work, how the result can be employed in MCN research and some of the challenges that remain unanswered by the literature. There exists an extensive body of work in the field of content and service placement in dis-



Mobile users workload | *Campus* workload | *Sporting event* workload

Figure 13: DC and Link utilisation CDF per placement method.

tributed compute and content delivery systems. Existing research results address many of the challenges facing an MCN.

7.1 Replica placement

There have been numerous efforts at developing algorithms for replicas in a network of computers [30]. In general, their objectives are to either optimize the application's performance in an existing infrastructure, to reduce the computational complexity of the decision, or minimize the infrastructure cost while meeting the application's SLA. The suboptimal algorithm proposed in [31] attempts to minimise the communication latency between the application and its clients by identifying and placing replicas in regions based on the relative latencies between nodes. Similarly, [32] proposes a topology-aware replica placement. The two algorithms achieve near optimal placement of replicas, but do not address the mobility of the service's users and thus the continuous evaluation of the service's placement in the network.

7.2 CDN and caching

Many challenges facing an MCN such as application placement in a distributed Telco-infrastructure and stochastic workload/demand are also found in CDNs. Research on CDNs has yielded mature centralised [33, 34] and distributed [35] methods for initial and churn-driven continuous placement of content in caching infrastructure core networks to mitigate network contention, and ensure content availability. The methods often employ a Mixed Integer Programming (MIP) approach to compute a global or local placement optima. The minimisation functions often incorporate system properties such as demand churn, network topology, and foreground traffic to meet the MNO's objectives. Due to the nature of CDN infrastructure the aforementioned methods do not take into account user mobility, application/content component affinity and execution, and finite heterogeneous resources.

7.3 Inter-and-Intra data centre VM-placement

Research in Intra-DC VM placement and continuous placement evaluation has yielded methods [36, 13, 37] to primarily consolidate applications, improve locality, and to minimise network and energy usage inside and across DCs. The literature often assumes a tree structured topology with resource contentions similar those found in an MCN. Methods that actuate Intra-DC VM placement are often agnostic to the intermediate topology between the DCs, relying primarily on observed performance. Although the surveyed methods adequately take into account network topology and heterogeneity, compute resources are assumed to be homogeneous, with no application component affinity. A high degree of resource heterogeneity and application component affinity are two fundamental properties of an MCN.

Again, the authors often resort to MIP to resolve a complex set of constraints and relationships between DC resources and application requirements. The objective is often to reduce/minimise Intra-DC network contention and to reduce cost and energy consumption by improving application locality and more appropriate VM to PM mappings. As the placement domain is either within a DC or in a set of DCs [38] the research fails to take into account end-user locality and rapid spatial and temporal demand changes in demand.

Research presented in [39] discern a method to increase individual end-user proximity to their user data by migrating and duplicating user instances on a planetary scale with the objective to reduce RTD. The work highlights the inherit challenges of what to migrate, duplicate, and replicate, and how to evaluate the action's performance return. The work does however not take into account demand churn and replicating whole instances of an application, nor does it consider a fine grained network topology, such as an MNO's access network. In [40], the authors have devised an approach to migrate and duplicate data across continental distances while taking into account

the intermediate network's availability and the contention the transfer incurs.

8 Conclusions

The highly distributed and heterogeneous nature of an MCN introduce several interesting autonomous resource management challenges arising from a highly dynamic workload, heterogeneous resources, rapid user mobility, heterogeneous energy costs, and multi-component applications. One of the foremost challenges in an MCN paradigm is how to manage the highly heterogeneous and distributed resources in a complex system. In this paper we introduces an objective function to minimise the global system cost as a means to manage the compute and network resources in an MCN. We have focused on static and continuous application placement methods, introducing an globally optimal placement scheme with a high computational cost and a locally optimal placement scheme with at a fraction of the computational cost. We developed an experimental simulation environment on which we evaluated the algorithms using a set of challenging MCN workloads.

The results reveal that the globally optimal and locally optimal schemes can achieve near equal performance with workloads that have spatially uniform distribution of demand. A difference in performance is revealed when subject to a workload with spatially non-uniform demand. Here, the globally optimal optimal outperforms the locally optimal. Nevertheless, with a transient workload, system cost and resource utilisation is with either algorithm non-divergent.

For future work, we would like to extend the experiments to include application component affinity to model the performance of for example multi component applications. Furthermore, we also intend to allow applications to dynamically replicate in the network. From an algorithm perspective, we would like to extend the current work to include a controller that regulates the rate of re-evaluation as to further minimise the computational complexity.

9 Acknowledgements

This work is funded in part by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control. Maria Kihl and William Tärneberg are members of the Lund Center for Control of Complex Engineering Systems (LCCC) funded by the Swedish Research Council (VR) and the Excellence Center Linköping - Lund in Information Technology (ELLIIT). William Tärneberg is also funded by the Mobile and Pervasive Computing Institute Lund University (MAPCI).

References

[1] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4:1–4:11, July 2009.

[2] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, First 2014.

[3] R. Shea, J. Liu, E.-H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *Network, IEEE*, vol. 27, no. 4, pp. 16–21, July 2013.

[4] P. Bosch, A. Duminuco, F. Pianese, and T. Wood, "Telco clouds and virtual telco: Consolidation, convergence, and beyond," in *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2011, pp. 982–988.

[5] M. Kihl, E. Elmroth, J. Tordsson, K.-E. Årzen, and A. Robertsson, "The challenge of cloud control," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, 2013.

[6] C. Adam and R. Stadler, "Service middleware for self-managing large-scale systems," *IEEE Transactions on Network and Service Management*, vol. 4, no. 3, pp. 50–64, Dec 2007.

[7] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[8] S. A. Baset, "Cloud slas: Present and future," *SIGOPS Oper. Syst. Rev.*, vol. 46, no. 2, pp. 57–66, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2331576.2331586

[9] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte wireless virtualization and spectrum management," in *Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP*, Oct 2010, pp. 1–6.

[10] S. Singh and P. Singh, "Key concepts and network architecture for 5g mobile technology," *International Journal of Scientific Research Engineering & Technology (IJSRET)*, vol. 1, no. 5, pp. 165–170, 2012.

[11] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella, "On the treeness of internet latency and bandwidth," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 61–72, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/2492101.1555357

[12] 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 *IEEE International Conference on Services Computing (SCC)*, July 2011, pp. 72–79.

[13] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1–9.

[14] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[15] A. ElNashar, M. El-saidny, and M. Sherif, *Design, Deployment and Performance of 4G-LTE Networks: A Practical Approach*, 1st ed. Wiley, 5 2014. [Online]. Available: http://amazon.com/o/ASIN/1118683218/

[16] C. Fraleigh, F. Tobagi, and C. Diot, "Provisioning ip backbone networks to support latency sensitive traffic," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, March 2003, pp. 375–385 vol.1.

[17] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1071690.1064252

[18] D. Schwab and R. Bunt, "Characterising the use of a campus wireless network," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2004, pp. 862–870.

[19] G. K. Zipf, "Human behavior and the principle of least effort." 1949.

[20] 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*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 241–252. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807166

[21] A. Mehta, J. Durango, J. Tordsson, and E. Elmroth, "Online spike detection in cloud workloads," in *2015 IEEE International Conference on Cloud Engineering (IC2E)*, March 2015, pp. 446–451.

[22] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration." in *USENIX Symposium on Networked Systems Design and Implementation*, vol. 7, 2007, pp. 17–17.

[23] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, Oct 2009, pp. 50–57.

[24] X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, "Dynamic load balancing for i/o-intensive applications on clusters," *ACM Transactions on Storage (TOS)*, vol. 5, no. 3, p. 9, 2009.

[25] S. Kundu, R. Rangaswami, K. Dutta, and M. Zhao, "Application performance modeling in a virtualized environment," in *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2010, pp. 1–10.

[26] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, "Autonomic placement of mixed batch and transactional workloads," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 219–231, Feb 2012.

[27] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling applications for virtual machine placement in clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 660–667.

[28] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, 2008.

[29] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[30] M. Karlsson and C. Karamanolis, "Choosing replica placement heuristics for wide-area systems," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.* IEEE, 2004, pp. 350–359.

[31] M. Szymaniak, G. Pierre, and M. van Steen, "Latency-driven replica placement," in *Proceedings of The 2005 Symposium on Applications and the Internet, 2005.*, Jan 2005, pp. 399–405.

[32] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384–392, 2002.

[33] ——, "Topology-informed internet replica placement," *Computer Communications*, vol. 25, no. 4, pp. 384 – 392, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366401004108

[34] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1587–1596 vol.3.

[35] S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1455–1468, Sept 2011.

[36] P. Svärd, W. Li, E. Wadbro, J. Tordsson, and E. Elmroth, "Continuous datacenter consolidation," ser. Report / UMINF, no. 2014:08. Umeå universitet, 2014, p. 12.

[37] F. P. Tso, G. Hamilton, K. Oikonomou, and D. Pezaros, "Implementing scalable, network-aware virtual machine migration for cloud data centers," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, June 2013, pp. 557–564.

[38] B. Kantarci, L. Foschini, A. Corradi, and H. Mouftah, "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *Globecom Workshops (GC Wkshps)*, Dec 2012, pp. 708–713.

[39] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services." in *USENIX Symposium on Networked Systems Design and Implementation*, 2010, pp. 17–32.

[40] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 74–85, 2011.