Future Generation Computer Systems 70 (2017) 163–177

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Dynamic application placement in the Mobile Cloud Network

William Tärneberg[a,*], Amardeep Mehta[b], Eddie Wadbro[b], Johan Tordsson[b], Johan Eker[c], Maria Kihl[a], Erik Elmroth[b]

[a] *Department of Electrical and Information Technology, Lund University, SE-223 63 Lund, Sweden*
[b] *Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden*
[c] *Ericsson Research, Lund, Sweden*

HIGHLIGHTS

- Proposes a holistic cost-optimal management approach for the forthcoming Mobile Cloud Network.
- The proposed method shows significant improvements in aggregate system cost and resource utilisation skewness, and proves to be very capable at accommodating user mobility.
- The evaluations reveal that high cost of user mobility and the amount of coordinated effort is required to accommodate it.
- The proposed method provides an upper bound to which we can contrast more tractable distributed solutions.

ARTICLE INFO

Article history:
Received 27 January 2016
Received in revised form
18 April 2016
Accepted 20 June 2016
Available online 20 July 2016

Keywords:
Mobile Cloud
Telco-cloud
Optimisation
Placement
Modelling
Distributed
Graph
Networks
Cloud computing
Edge
Mobile
Infrastructure

ABSTRACT

To meet the challenges of consistent performance, low communication latency, and a high degree of user mobility, cloud and Telecom infrastructure vendors and operators foresee a Mobile Cloud Network that incorporates public cloud infrastructures with cloud augmented Telecom nodes in forthcoming mobile access networks. A Mobile Cloud Network is composed of distributed cost- and capacity-heterogeneous resources that host applications that in turn are subject to a spatially and quantitatively rapidly changing demand. Such an infrastructure requires a holistic management approach that ensures that the resident applications' performance requirements are met while sustainably supported by the underlying infrastructure. The contribution of this paper is three-fold. Firstly, this paper contributes with a model that captures the cost- and capacity-heterogeneity of a Mobile Cloud Network infrastructure. The model bridges the Mobile Edge Computing and Distributed Cloud paradigms by modelling multiple tiers of resources across the network and serves not just mobile devices but any client beyond and within the network. A set of resource management challenges is presented based on this model. Secondly, an algorithm that holistically and optimally solves these challenges is proposed. The algorithm is formulated as an application placement method that incorporates aspects of network link capacity, desired user latency and user mobility, as well as data centre resource utilisation and server provisioning costs. Thirdly, to address scalability, a tractable locally optimal algorithm is presented. The evaluation demonstrates that the placement algorithm significantly improves latency, resource utilisation skewness while minimising the operational cost of the system. Additionally, the proposed model and evaluation method demonstrate the viability of dynamic resource management of the Mobile Cloud Network and the need for accommodating rapidly mobile demand in a holistic manner.

1. Introduction

Contemporary cloud services are predominantly hosted by a handful of Data Centres (DCs) on each continent [1], with no specific regard to their subscribers' whereabouts.

Furthermore, cloud services are at an increasing rate accessed from Mobile Devices (MDs) [2] and employed to orchestrate the infrastructure and devices that constitute the Internet of

* Corresponding author.
E-mail addresses: william.tarneberg@eit.lth.se (W. Tärneberg), amardeep@cs.umu.se (A. Mehta), eddiew@cs.umu.se (E. Wadbro), tordsson@cs.umu.se (J. Tordsson), johan.eker@ericsson.com (J. Eker), maria.kihl@eit.lth.se (M. Kihl), elmroth@cs.umu.se (E. Elmroth).

http://dx.doi.org/10.1016/j.future.2016.06.021
0167-739X/© 2016 Elsevier B.V. All rights reserved.

Everything (IoE) [3]. Collectively, they produce and consume an increasing amount of content, consuming a larger portion of network traffic. Due to intermittent cloud service availability and heterogeneous latency between an MD and a DC, MDs often resort to executing applications locally, as opposed to in the cloud [4].

The current, centralised infrastructure paradigm, geographically separates users and infrastructure, and thus does not accommodate mission critical, latency sensitive application, and incurs a significant load on the network. The Mobile Cloud Network (MCN) [5] has been proposed to mitigate these performance inhibitors. In the MCN paradigm, DCs of heterogeneous scale and cost are dispersing throughout the core- and access-networks as a compliment to the existing, centralised, public cloud DCs [6], see Fig. 1. The proposed infrastructure will span across entire national networks, encompass thousands of subscribers, nodes, and applications. The compute capacity, or DCs, will proposedly reside with a varying degree of proximity to the end-user, from Radio Base Stations (RBSs) to regional hubs. Such an infrastructure will be both cost- and capacity-heterogeneous as a function of depth. Furthermore, the proposed use case of the MCN ranges from hosting virtualised Telecom service [7], offloading mobile application [8], mobile edge computing [9,10], to orchestrating and enabling Internet of Things (IoT) services [11,12].

With increased proximity between end-users and cloud infrastructure, application developers and device manufacturers are now able to realise MD offloading and large Wireless Sensor Network (WSN). As a consequence, they are for example faced with the challenge of determining when and what to offload [13], given capacity and energy objectives [14], and how to design their sensor infrastructure [15–18].

Furthermore, regardless of how application owners and developers take advantage of the MCN, to realise an MCN, the operator of an MCN shall attempt to meet all resident applications' performance goals and to manage the operational expenditure of its infrastructure. In such a large, distributed, cost-, proximity-, and capacity-heterogeneous infrastructure, with large set of highly mobile users, the placement of the resident applications and services is an MCN operator's foremost degree of freedom. The operator of an MCN thus pursues its objectives by actively evaluating the placement of the resident applications in relation to each other, their mobile users, and cost and capacity of the infrastructure's resources. The fundamental problem addressed in the paper is how to scalably and autonomously manage the infrastructures in such a manner, over a large number of resources, to guarantee application performance, while minimising the incurred cost on the infrastructure given a time-variant demand and a heterogeneous infrastructure, and mitigating resource skewness.

If the resident applications' placements are not evaluated at the rate of which the applications' demand is changing, the system cannot guarantee the applications' performance goals or Service Level Agreements (SLAs), or minimum operational cost or system wide resource usage. The task of evaluating the placement of thousands of applications serving an even greater number of users across hundreds of heterogeneous nodes cannot be done manually, in real-time, by a system administrator. Furthermore, the breadth and depth, and heterogeneity of the infrastructure, and the rate of change in application demand make judging the trade-off, over time, non-trivial, and need to be performed holistically and autonomously by the system.

Restoring to naïve methods, such as placing applications randomly or greedy placing applications as close as possible to the end-user, fails to take into account the time-variant input to the system and the cost- and capacity-heterogeneity of the system, and can thus not guarantee that the applications meet their SLAs, nor does it guarantee that the cost of the system is maintained, and that the load is relatively balanced across the system.

As the MCN paradigm is novel, no implementation exists nor has a full definition materialised. As such, this problem has to the best of our knowledge not been addressed previously. Related performance centric problems can be found in literature for routing, intra- and inter-DC application placement, and in optimal content distribution in Content Delivery Networks (CDNs). However, to the best of our knowledge, none of the presented approaches simultaneously and holistically take into account rapid user mobility and vast resource cost- and capacity-heterogeneous infrastructures.

The contribution of this paper is threefold. (1) A performance model is presented of the proposed MCN infrastructure paradigm. (2) An optimisation problem based on aforementioned objectives is formulated and solved. The optimal solution achieves a 25% reduction in cost compared to the naïve methods. (3) A traceable, iterative, local-optimal algorithm is presented. This algorithm achieves a cost within 5% of the optimal, at only a fraction of the computational overhead.

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 the problems as an optimisation problem. Section 5 describes application placement methods based on the objective function described in the earlier section. Sections 6 and 7 present how the proposed algorithm and models are evaluated by detailing the evaluation model and the experiments, respectively. Section 9 covers related work in tangent research fields. Section 10 summarises this paper's contributions, conclusion of the experiments and future work.

2. Resource management challenges

This section provides a definition of the MCN and formalisation of an MCN's service offerings. Thereafter, its resource management objectives are defined and the inherit resource management challenges are discussed.

2.1. Infrastructure

In contrast to today's prevailingly centralised cloud infrastructure, an MCN employs a distributed resource- and cost-heterogeneous infrastructure embedded into the core and access networks [5,7], see Fig. 2. DCs in an MCN supply compute, memory, storage, and network capacity. The intermediate access- and core-networks, and the Radio Access Network (RAN), provide the means to access these DCs, and haul traffic between the resident DCs and backbone networks beyond the MCN.

In terms of the distributed DCs, compute and network capacity diminishes with depth, while the per unit time cost for its resources increases towards the leafs of the network. The DCs located at the edge of the access network, for example, adjacent to an RBS or in an office are assumed to be of much smaller scale than the large distant DCs. The smaller edge-DCs offer low latency compute capacity or specialised services at a higher relative operational cost. Large distant DCs are driven by economy of scale and provide generic low-cost compute resources, at a higher latency. For example, an application placed at the edge of the network will have a lower Round Trip Time (RTT) to its end-users, less aggregate traffic, but executed at a higher operational cost as compared to a traditional distant data centre.

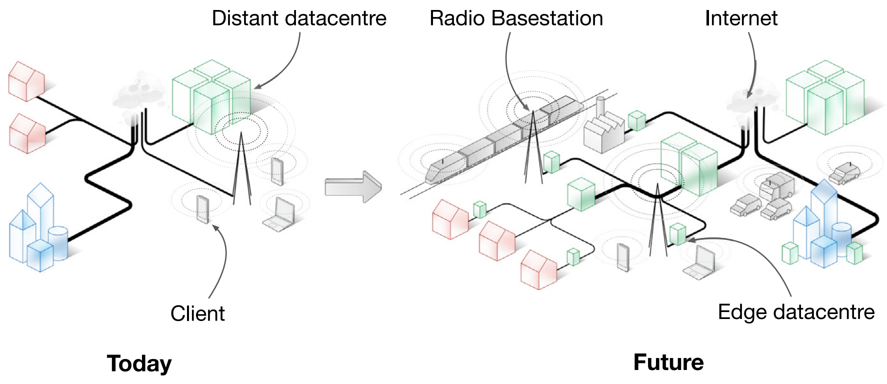


Fig. 1. Overview of the present cloud topological paradigm versus the MCN.

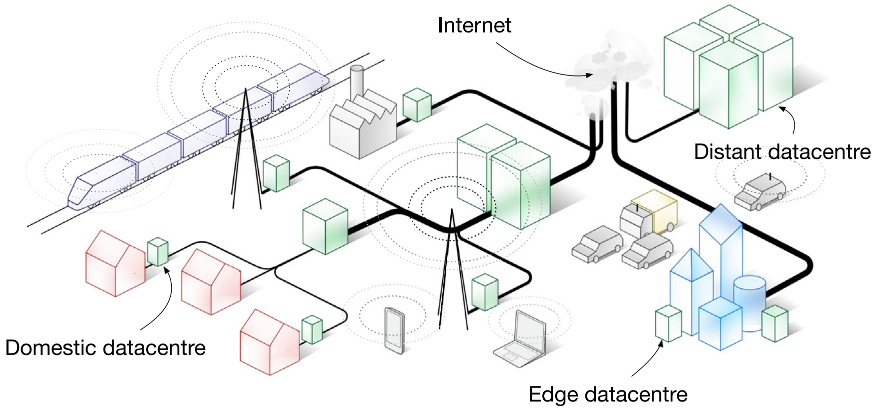


Fig. 2. An MCN with a distributed DCs in the core- and access-networks.

2.2. Application and service paradigm

Applications are submitted to an MCN by their owners from beyond and within the network to serve a subset of the network's users. The decision to submit an application to an MCN is assumed to be made by the application owner based on the performance and cost merits of the MCN. The resources in an MCN can be viewed as resources in a federated cloud [19] that are brokered globally but are, for example, sought after for their locality to a specific group of users, specialised sources, sensors, and/or actuators. Although cloudlets [20] might be a part of the composition of applications, they are not distinguished from the rest of the application set.

An application in an MCN serves a set of users. The users can be stationary or mobile. An application can for example serve just one user in the network, a small set of distributed users, or a large set of users dispersed throughout the network.

Application owners impose performance, availability, latency, and locality requirements on an MCN in the form of Service Level Objectives (SLO) or a Service Level Agreement (SLA) that the operator of the MCN is expected to comply with. However, both end-users and application owners alike are agnostic to where an application is hosted in the infrastructure and how the network and the cloud infrastructures are managed. It is assumed that the end-users cannot impose requirements on the network or on the applications' performance. It is also assumed that there is no resource competition between applications and that an MCN does not honour priorities. A cloud-based application is assumed to be contained within a Virtual Machine (VM) or a container.

From this point on, the software constituting a service running in a DC is referred to as an application, and the relationship between the end-users and the application as a service. An application's SLO or SLA is referred to as its performance goals.

2.3. Resource management objectives

In this work, the holistic orchestration of an MCN's resources and the applications it hosts is referred to as resource management. I.e. the operator of a Mobile Network Operator (MNO) employs a resource management principal to meet its performance objectives.

It is assumed that an MNO's overall resource management objectives are to ensure that the fundamental circumstances for all applications to perform according to their individual SLO or SLA are met while the system-wide operational costs are minimised. Further, it is also assumed that an MCN and its resident Telecom services in the network are resource managed independently. The MCN's resource management policies are also assumed to be symbiotic with the network's resource management policies, in the sense that it contributes to the objectives of the network by for example reducing resource utilisation of the underlying access network. Furthermore, it is assumed, that minimising resource utilisation and resource utilisation skewness, increases the availability of the infrastructure.

An MCN operator can practice admission control and is thus able to reject new applications if they violate its internal management objectives and the integrity of the other application's SLO or SLA. However, this work focuses solely on solving on-line resource management to maximise the availability of the infrastructure, and thus only deals with admitted, resident applications. Once an application has been admitted and placed in a DC, the performance of an application is the result of the application's resource usage properties, the properties of the DC it is running on, and its relative location to its demand/end-user(s), orchestrated by the operator.

Furthermore, the internal management policies and practices of the DCs, such as the execution of applications, are beyond the scope of this work. Therefore, VM or application to Physical Machine (PM) mapping or application placement in a DC does not fall within the premise of this paper. Furthermore, applications hosted by an MCN are assumed not to have a scheduled deadline, but are rather terminated based on the application's internal management objectives.

It is assumed that an MNO's MCN is managed on top of the existing Telecom infrastructure. The MCN management process is agnostic to 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 host the admitted applications with a finite set of resources.

In conclusion, the resource management objectives of an MCN operator can be summarised as:

- Meet all resident applications' SLAs.
- Minimise infrastructure-wide operational expenditure.
- Load balance of resources and mitigate resource usage skewness.

2.3.1. *Degrees of freedom*

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 of 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 heterogeneity.
- Which pieces of infrastructure to run.
- Where to run the applications.

Out of these four, on-line or continuous evaluation of the admitted applications' placement is the decision variable with the greatest ability to meet the system's management objectives. It is assumed that an application can consume any amount of resources and be subjected to a demand, in any manner in the realm of what is physically and computationally possible. An operator of an MCN thus needs to accommodate these changes and manage its infrastructure accordingly.

2.4. *Challenges*

The sheer size of the infrastructure and the number of resource management parameters render a fully centralised resource allocation strategy practically infeasible [21]. A centralised system will be able to provide an optimal resource management solution but might be practically infeasible due to, for example, computational complexity, scalability, and fault tolerance. Consequently, a decentralised collaborative resource management approach is desired.

Where to place the application within that set will be a trade-off between the needs of the other resident applications, the available resources within the set of possible placements, and the cost of running the application in each of those choices. Each application has a set of resource and performance requirements that determine where it can run and how much it will cost in each DC. Because of an application's performance requirements and size, not all applications can be run in all DCs. Due to the cost- and capacity-heterogeneous nature of an MCN, independently and greedily placing each application where they minimise the mean distance to its end users does not guarantee that is the placement that incurs the least cost. Nor does it guarantee that the placement maximises the utility of the infrastructure's resources. The objective of an MCN's operator is to meet each application's performance objectives, not exceed them, which yields a larger set of possible application placements throughout the breadth and depth of the infrastructure. Furthermore, what might be considered an ideal placement for an application might change over time as its demand moves and needs for other applications have changed. An MCN's resources are limited and need to be managed so that it can handle changes in demand and needs for its resident applications. Managing these trade-offs at the rate of change in the system and at the scale of the system can feasibly not be achieved manually by a system administrator.

In conclusion, operating a functioning MCN relies on an operational network and the ability to cost-manage that network. The resource heterogeneity of an MCN infrastructure, the mobility of its resident users and the heterogeneity of the applications to which they subscribe, introduce a complex set of resource management decisions. The application placement algorithm presented in the paper solves this management challenge in a holistic manner by taking into account both operational cost of the infrastructure, the utilisation of the infrastructure, and the individual performance of each application.

3. Proposed system model

In order to explore the properties of an MCN and the management challenges it presents, a system model of the MCN is proposed. This section begins with the constitution of 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 Fig. 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 [7]. The placement and scale of an MCN's DCs will be heavily dependent on the degree of an MNO's infrastructure virtualisation [22], 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 have not yet solidified [23]. Additionally, this work makes no specific assumptions about the placement or scale of the infrastructure. However, the models are generic enough to handle a number of feasible next-generation infrastructure topologies.

An MCN is modelled as an undirected forest or tree graph [7,24–26], where the vertices are DCs and the edges are network links, each with a set of finite resources, see Fig. 3. Furthermore, applications are hosted in DCs and are subject to a demand through the network links, originating at the leaf nodes. The graph $G = (V, E)$ denotes a tree depicting the MCN's network topology, where

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

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 follow

$$\mathrm{dist}(v_k, v_1) \le \mathrm{dist}(v_i, v_1), \forall k \le i, \tag{2}$$

where the distance between nodes $v \in V$ and $w \in V$, $\mathrm{dist}(v, w)$ is the number of vertices that is on the shortest path from v to w.

The leaf vertices are connected by RBSs from which the end-users access the network. The leaf vertices function as geographic aggregation points for the applications' demand.

3.1. *Data center model*

An MCN's compute capacity is provided by a set of cost- and capacity-heterogeneous DCs distributed at various depths of the network. The capacity is assumed to decrease with depth, while resource per unit cost is assumed to increase with depth.

Each vertex is a DC that is able to host applications that use a finite set of compute and network resources. Vertex v_i, $i \in \{1, 2, \dots, I\}$ in the graph has the following features.

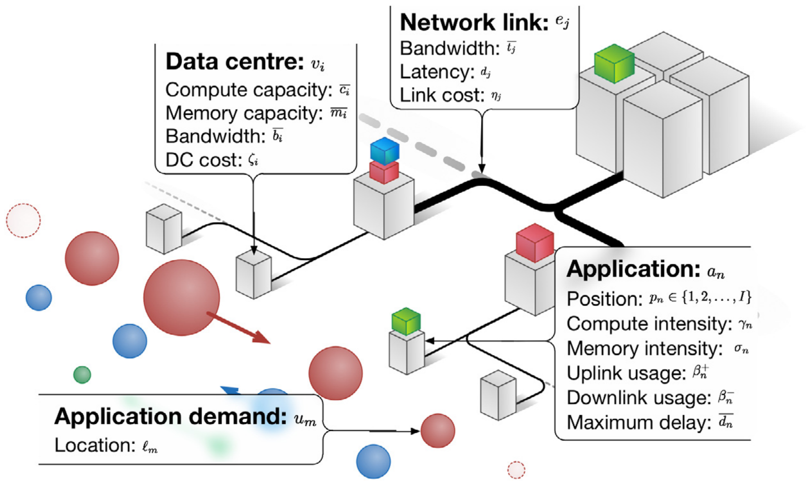

Data centre: v_i
Compute capacity: $\overline{c_i}$
Memory capacity: $\overline{m_i}$
Bandwidth: $\overline{b_i}$
DC cost: ζ_i
Network link: e_j
Bandwidth: $\overline{t_j}$
Latency: d_j
Link cost: η_j
Application: a_n
Position: $p_n \in \{1, 2, \ldots, I\}$
Compute intensity: γ_n
Memory intensity: σ_n
Uplink usage: β_n^+
Downlink usage: β_n^-
Maximum delay: $\overline{d_n}$
Application demand: u_m
Location: ℓ_m

Fig. 3. Model overview.

- **Compute capacity** $\overline{c_i}$, a number describing the total compute capacity of 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

- **DCcost** ζ_i, a function of resource usage (compute 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 [27].

3.2. Network model

Existing 3rd and 4th generation mobile access networks are generally tree-structured [28]. 5th generation networks are also expected to follow the same structure. 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, $j \in \{1, 2, \ldots, J\}$, in the graph has a network resource capacity expressed as bandwidth $\overline{t_j}$, i.e. a number specifying the maximum throughput over the edge. In addition, each edge has a link cost η_j, i.e. a function of throughput that returns the link's running cost per time unit. The bandwidth cost is assumed to increase with the distance to the root vertex.

The communication latency of an application is dictated and maintained by the application's relative location 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 an application's demand can vary with time.

3.3. Application model

An MCN hosts applications a_n, where $n = 1, 2, \ldots, N$, in one DC. Let $A = \{a_n \mid n = 1, \ldots, N\}$ denote the set of all applications hosted by an MCN. Application a_n, $n \in \{1, 2, \ldots, N\}$, has the following features.

- **Position** $p_n \in \{1, 2, \ldots, I\}$, a number specifying that application a_n is running on the DC at vertex v_{p_n}.
- **Compute intensity** γ_n, an increasing function of the demand of application a_n that describes the amount of computational resources required by the application component.
- **Uplink usage** β_n^+, an increasing function of the demand of application a_n as well as the location of the application's end-users that returns the uplink throughput associated with the application.
- **Downlink usage** β_n^-, an increasing function of the demand of application a_n as well as the location of the application's end users that returns the downlink throughput associated with the application.

As a result, applications scale vertically in a DC, as a function of demand.

3.4. User model

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

- **Location** $\ell_m \in \{1, 2, \ldots, I\}$, a number specifying that demand source u_m is currently served by the DC at vertex location l_m.
- **Active application** $A_m \subset A$, the application serving the demand from u_m.

For future notation, $U_n = \{u_m \in U \mid n \in A_m\}$ is defined as the demand for an application n and let $U_{n,i} = \{u_m \in U_n \mid \ell_m = i\}$.

4. Optimisation formulation

In this section, a set of application placement objectives for resource managing an MCN is formalised from which an optimal application placement algorithm is derived. This section begins by introducing a cost function reflecting the presented management objectives, encompassing the operational cost of an MCN's sources and a heuristic overload cost. The optimal placement algorithm minimises this function over all feasible placement constellations. A local-optimal algorithm is then derived and formulated.

4.1. Resource metrics and constraints

The optimisation objective in this paper is to decrease the running costs of an MCN by placing/moving the resident applications at/between different DCs to minimise the operational cost of the infrastructure's DC resources and minimising 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, let the set of admissible placements be

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

Below, the resource metrics at the vertices (DCs) and edges (links) are defined, which can be computed from the features of the four models introduced in Section 3. To detail the relations, 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 sources of demand and applications such that the link is on the path connecting the DC serving the demand and the DC hosting the application. More precisely, the throughput is given by

$$t_j = \sum_{\{n,i \mid e_j \in P_{p_n,i}\}} \left(\beta_n^+(U_{n,i}) + \beta_n^-(U_{n,i}) \right), \tag{4}$$

where $n \in \{1, \dots, N\}$, $i \in \{1, \dots, I\}$.

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

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

The compute usage at DC i is the sum of the corresponding usage by the applications that are running at the DC, that is,

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

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

$$b_i = \sum_{\{j \mid e_j \in E_i\}} t_j. \tag{7}$$

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

$$c_i/\overline{c_i} \le 1, \qquad i = 1, 2, \dots, I, \tag{8}$$
$$b_i/\overline{b_i} \le 1, \qquad i = 1, 2, \dots, I, \tag{9}$$
$$t_j/\overline{t_j} \le 1, \qquad j = 1, 2, \dots, J, \tag{10}$$
$$d_{n,m}/\overline{d_n} \le 1, \qquad n = 1, 2, \dots, N \text{ and } \{m \mid u_m \in U_n\} \tag{11}$$

where $\overline{c_i}$ and $\overline{b_i}$ are the total compute and bandwidth capacity of DC i respectively, $\overline{t_j}$ is the maximum throughput for link j and $\overline{d_n}$ is the maximum allowed latency for application a_n. All constraints above are written in the form $a/\overline{a} \le 1$, the relative usage of a particular resource should thus 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

The objective function in Eq. (12) captures 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 provider's viewpoint, whose objective is to minimise the overall running cost

$$J(\mathbf{p}) = \sum_{i=1}^{I} \zeta_i(c_i, b_i) + \sum_{j=1}^{J} \eta_j(t_j). \tag{12}$$

It is assumed that the cost for running the DCs is linearly proportional to their compute resource usage and that the cost for the network is linear with respect to the throughput over each link. Remark that adding a constant background cost that represents the cost for when the links and DCs are idle does not influence the possible savings by migrating the applications. The optimisation problem is formulated as

$$\min_{\mathbf{p} \in \mathcal{A}} J(\mathbf{p}), \tag{13}$$

subject to constraints (8)–(11).

The problem formulation above is straightforward and intuitive. However, in this system the demand for an 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, a penalty initialisation point $\tilde{x} < 1$ is introduced which defines the penalty–barrier function

$$f_{\tilde{x}}(x) = \begin{cases} 0, & \text{if } x < \tilde{x}, \\ \dfrac{1}{1-x} + \dfrac{2\tilde{x} - x - 1}{(1-\tilde{x})^2} & \text{if } x \ge \tilde{x}. \end{cases} \tag{14}$$

In the equation above, x should be viewed as the relative usage (or relative latency), more precisely the quotient on the left hand side in constraints (8)–(11). For the case when $x \ge \tilde{x}$, the first term is selected to ensure that $f_{\tilde{x}}(x) \to \infty$ as $x \to 1$ and the second term is selected 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 $x \le \tilde{x}$ and a barrier function for the constraint $x \le 1$. In essence, this makes it easy to modify the point where the penalisation starts for each constraint separately. The algorithm utilises 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 where the corresponding DC is located. To compute the overall penalty G, the penalty–barrier function is added, corresponding to all constraints with their respective penalty initialisation points. That is,

$$G(\mathbf{p}) = \sum_{i=1}^{I} f_{\tilde{c}}\left(c_i/\overline{c_i}\right) + \sum_{i=1}^{I} f_{\tilde{b}}\left(b_i/\overline{b_i}\right) + \sum_{j=1}^{J} f_{\tilde{t}}\left(t_j/\overline{t_j}\right)$$
$$+ \sum_{n=1}^{N} \sum_{\{m \mid m \in U_n\}} f_{\tilde{d}}\left(d_{n,m}/\overline{d_n}\right), \tag{15}$$

where $\tilde{c}$, $\tilde{b}$, $\tilde{t}$, and $\tilde{d}$ are the penalty initialisation points corresponding to the constraints for compute usage, DC throughput, link throughput, and application latency, respectively. Here, an individual penalty–barrier function is added for all constraints. This corresponds to taking the 1-norm of the vector that holds 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 (13) subject to constraints (8)–(11), the following problem is solved

$$\min_{\mathbf{p} \in \mathcal{A}} J(\mathbf{p}) + \mu G(\mathbf{p}), \tag{16}$$

where μ is a positive penalty parameter and J and G are defined in expressions (12) and (15). 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

In this section, two methods to re-evaluate and place applications in an MCN using the optimisation formulation and constraints are presented. The first method is an exhaustive search algorithm that finds the optimal placement and serves as a performance upper bound reference. The second method is a tractable iterative local search algorithm that operates on the same optimisation premise as the exhaustive method. The objective of the iterative method is to provide a tractable centralised solution that converges to a near-optimal solution.

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

$$\mathcal{N}_{\mathbf{p}}^{k} = \{\boldsymbol{q} \in \mathcal{A} \mid \|\boldsymbol{p} - \boldsymbol{q}\|_{\mathcal{A}} \leq k\}, \tag{17}$$

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, a k such that $|\mathcal{N}_{\mathbf{p}}^{k}|$ needs to be selected so that it is not too large. The network distance can be computed as,

$$\|p - q\|_{\mathcal{A}} = \sum_{n=1}^{N} \delta_{\mathcal{A}}(p_n, q_n), \tag{18}$$

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

$$\delta_{\mathcal{A}}(p_n, q_n) = \begin{cases} 1, & \text{if } p_n \neq q_n, \\ 0, & \text{else} . \end{cases} \tag{19}$$

An alternative choice is

$$\delta_{\mathcal{A}}(p_n, q_n) = \begin{cases} |P_{p_n,q_n}|, & \text{if } p_n \neq q_n, \\ 0, & \text{else} , \end{cases} \tag{20}$$

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

5.1. Exhaustive search

To yield the optimal solution to problem (16), so-called exhaustive search is performed, that is to evaluate the objective function for each admissible placement. Doing so provides an upper performance bound of the system at each time it re-evaluates the system. However, the computational complexity of this approach is exponential, since $|\mathcal{A}| = N^I$. Thus, this approach is not feasible for large-scale systems.

The algorithm re-evaluates the objective function, $J + \mu G$, during run-time to compute the total cost for running all the applications inside the infrastructure. With a priori knowledge of the system's rate of change one could adjust the rate at which the applications' placements are re-evaluated, accordingly. However, the rate or re-evaluation is beyond the scope of this work, therefore, in this paper, the algorithm is designed to re-evaluate periodically.

As applications are admitted to the system, an initial placement is performed by searching for the globally optimal location for all applications, which minimises the objective function detailed in Section 4. In other words, there is no formal distinction between initial and continuous placement.

5.2. Iterative local search

To make the algorithm more scalable, the number of evaluations can be reduced by limiting the spatial search domain for each application, in the following called the Iterative local search approach. This algorithm employs a depth first search approach to find the neighbourhood nodes in the tree. The algorithm starts by finding all the solutions with neighbourhood depth k and computes the local optimal solution. From the local optimal solution, the algorithm constructs its subsequent k-neighbourhood and computes a new optimal solution. The algorithm then repeats the process until it achieves the specified maximum number of iterations or a locally optimal solution is found. The algorithm is described as in Algorithm 1.

Algorithm 1 Iterative local search algorithm

```
1:  Input : current placement vector of applications ($\mathbf{p}$), maximum
    number of iterations (maxIter), depth ($\mathbf{k}$)
2:  Output : new placement vector for applications
3:  $C_{\mathbf{p}} \leftarrow J(\mathbf{p}) + \mu G(\mathbf{p})$
4:  nIter $\leftarrow 0$
5:  while nIter $<$ maxIter do
6:      $\mathcal{N} \leftarrow$ get $\mathcal{N}_{\mathbf{p}}^{k}$, the $k$-neighbourhood of $\mathbf{p}$
7:      $C_{\mathbf{q}} \leftarrow \min_{\mathbf{q} \in \mathcal{N}} J(\mathbf{q}) + \mu G(\mathbf{q})$
8:      if $C_{\mathbf{p}} \leq C_{\mathbf{q}}$ then
9:          break
10:     end if
11:     $C_{\mathbf{p}} \leftarrow C_{\mathbf{q}}$
12:     $\mathbf{p} \leftarrow \mathbf{q}$
13:     nIter $\leftarrow$ nIter $+ 1$
14: end while
15: return $\mathbf{p}$
```

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.

Each application's distance to its optimal placement determines over how many iterations the algorithm will converge. It is therefore important to have a strategy for placing admitted applications. In the simplest form, the algorithm can for example start by randomly placing applications in the network and then let the placement algorithm converge over a certain number of iterations. Starting in a random location can produce situations where the application's placement might never converge to its optimal point. In this paper, the algorithm places applications as they are admitted in an optimal manner by doing an exhaustive search of the entire network.

6. Evaluation model

In this section, the simulation set-up is detailed. The behaviour of the placement algorithms is studied by subjecting them to a variety of configuration and workload scenarios. The performance of each algorithm is evaluated using a set of heuristics.

Section 6.1 outlines the premise of the experiments and their objectives. Section 6.2 details the simulation software framework used to execute the experiments.

6.1. Evaluation method

This section describes the experiment platform on which to study the properties and behaviours of the algorithms. First, a description of the different degrees of freedom of the experiments is provided, such as the simulator's parameters, workload properties, and the parameter space for the placement algorithms.

To identify the system's performance boundaries, the placement of each application is evaluated at run-time in a discrete time manner, employing a set of placement algorithms to regularly re-evaluate the placement of all applications to minimise total system cost of the three-structured system. To evaluate the performance of the placement methods, a number of performance metrics are observed, including the overall heuristic system cost, resource utilisation, and application RTT. Primarily, the overall cost as defined in Eq. (12), will contrast how well each placement algorithm deals with a specific scenario or configuration, and thus the system's operational objectives. Secondly, the resource utilisation will show 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.

This section proceeds by detailing the ingress workload to the system in Section 6.1.1. Sections 6.1.2 and 6.1.3 detail the infrastructure topology and the types of applications modelled, respectively.

6.1.1. Application demand

The application demand model is constructed using four parameters that enable 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, three relevant example scenarios are considered. The first scenario has a high degree of *User Mobility*, where demand for each application is concentrated to a few leaf nodes and is highly mobile. Second, a diurnal-centric *University Campus* scenario is presented, where demand of all applications is concentrated at the University campus during day time and disperses geographically to a uniform distribution at night. Furthermore, the 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, the spatial demand distribution is modelled as a normal distribution over a 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. The amount of demand is not varied, but the spatial distribution changes with time among the nodes having the demand for the application. Furthermore, the popularity distribution among applications is modelled using a uniform distribution. 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 applications such as Augmented Reality, autonomous vehicles applications, or cloudlets.

In the *University Campus* scenario, the demand for an application across the network's leaf nodes is modelled as a normal distribution. To capture the students' pronounced diurnal migration patterns [29], the standard deviation (σ) is varied diurnally to achieve a night-time 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. The popularity among applications is modelled using a Zipf distribution [30,31]. The demand varies from a uniform distribution during night to a normal distribution 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.

An unexpected surge in demand of these applications is modelled as a spike. In previous work [32], spikes were formally analysed and defined as a significant shift in the workload pattern. Here, in this work the time variation for the growing phase of a spike is modelled as,

$$y_\tau = y_{\tau-1}(1 + \alpha_g(1 - \lambda_g y_{\tau-1})), \tag{21}$$

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. The inverse of λ is the maximum of the popularity. The decreasing phase of a spike is modelled as,

$$y_\tau = y_{\tau-1}(1 - \alpha_d(\lambda_d y_{\tau-1} - 1)), \tag{22}$$

where α_d and λ_d are coefficients. Using the model defined above, a spike similar to the one experienced during Fifa 1998 world championship [33] is constructed 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 phases for the second peak. The spike is only present in one node, and is spatially stationary.

6.1.2. Infrastructure and topology

In the experiments, the infrastructure is composed of a set of DCs, distributed amongst the veracities of a tree with a certain depth. The capacity and cost of the DCs vary with depth. The veracities are connected with links. The links' capacities progressively diminish with depth.

Application demand originates from the leaf nodes in the network and it propagates to the DC where the application is hosted. Application demand incurs a proportional resource usage on the DC its applications are hosted on and on the intermediate links.

For each workload scenario, the total amount of resources required for all the applications at each leaf node at each time instance is computed. Based on the aggregate demand, the nodes and links of the system are allocated a proportional amount of resources, so that no application is denied admission to the system due to lack of resources. The remaining nodes are allocated a proportion of the aggregate demand from its children.

6.1.3. Application types

Applications consume heterogeneous resources proportional to the ratio of resources consumption profile and the size of its demand, per resource type. In the presented model, applications can be CPU, or I/O intensive [34–38]. An example of a CPU intensive benchmark application is Sysbench, similarly PostMark is a benchmark for I/O intensive applications.

Since the infrastructure is cost- and capacity-heterogeneous, each application type is expected to have a bias towards a certain depth of the network where it incurs the least operational cost globally or locally. For example, I/O intensive applications are likely to gravitate towards the capillaries of the network where they incur less aggregate network traffic and where I/O is relatively cheaper.

6.1.4. Placement algorithm parametrisation

In this section the exhaustive- and local-search algorithms are parametrised and details are provided on how they are configured in the experiments. The experiments include the following placement principals/algorithms:

- **Random static**, where each application component is initially placed at random in the network and is not dynamically moved.
- **Random continuous**, where each application at each re-evaluation 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 is relocated accordingly. The search depth is set to 4.

The methods that attempt to minimise the global cost are bound by the parameters of the cost function and on the penalty function defined in Eq. (14) and can be configured in a number of different manners to accommodate different objectives. The optimisation function equation (16) contains a weight μ that combines the overall running cost and the overall penalty (or overload cost). The Iterative local search was specified in Section 5.2. This algorithm minimises the cost function specified in Section 4, iteratively, over a confined neighbourhood k, a maximum of **maxIter** times per evaluation.

6.2. Simulator

An event driven simulator is used to validate and evaluate the topology and the presented placement methods. Existing simulation frameworks such as NS-3 [39] and CloudSim [40] offer competent network and data centre models, respectively. Nevertheless, neither frameworks offer a complete solution at the desired abstraction level nor do they scale adequately for large networks. As a result, a coarse-grained event-driven simulator in Python was developed. The simulator utilises SimPy [41] as the underlying event-driven framework. Furthermore, the simulator is constructed around the system 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 composed of a quantity of demand for each application in each leaf node for each time instance. The workload is propagated throughout the network to the DCs in which the application is hosted, incurring a resource usage on the host resources proportional to the demand.

To manage the experiments and its outputs the simulator features a centralised management unit that places new applications as they arrive and updates the network with resources consumed by the applications. 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 5. The simulator monitors and outputs the momentary total cost, application RTT, and resource utilisation levels.

7. Experiments

This section details workload scenarios and parameter settings.

7.1. Workload scenarios

Three workload scenarios are presented, Mobile user scenario, University campus and Sporting event. For the *Mobile user* scenario, three nodes are selected at random and users are distributed amongst them according to a normal distribution ($\mu = 20, \sigma = 10$). The demand is spatially moving according to a random walk.

For the University campus scenario, users are distributed among all the leaf nodes according to a normal distribution. The mean and standard deviation of the distribution are varied with time as shown in Table 1. A histogram is then generated with bin sizes equal to number of leaf nodes, 6 in this paper. The time stamps 1 and 5 represent mid-night and mid-day workload for the university campus scenario respectively.

For the sporting event scenario, application popularity is initially uniformly distributed with parameters $a = 0.1, b = 0.6$. An application is made popular by assigning a higher probability after time stamp 25. Parameters $\alpha = 0.4, \lambda = 0.001$ are used for Eqs. (21) and (22).

Table 1
University campus scenario parameters.

Time instance	Parameters
0	$\mu = 10, \sigma = 50, size = 100$
1	$\mu = 15, \sigma = 40, size = 100$
2	$\mu = 20, \sigma = 30, size = 100$
3	$\mu = 30, \sigma = 20, size = 100$
4	$\mu = 40, \sigma = 10, size = 100$
5	$\mu = 50, \sigma = 1, size = 100$

Table 2
Topology and infrastructure parameters.

	Property	Value	Description
Topology	Depth	3	Network depth
	DCs	9	Number of DCs in the network
	Links	16	Number of links in the network
DC capacity	Large	1	Large DC prop. capacity (Reference)
	Medium	$\frac{1}{2}$	Medium DC prop. capacity to Large DC
	Small	$\frac{1}{6}$	Small DC prop. capacity to Large DC
DC cost	Large	1	Large DC prop. cost (Reference)
	Medium	$\frac{5}{9}$	Medium DC prop. cost to Large DC
	Small	$\frac{2}{9}$	Small DC prop. cost to Large DC
Link capacity	–	–	Link capacity is homogeneous
Link cost	–	–	Link cost is homogeneous

Table 3
Heterogeneous application Model.

Type	Compute (%)	IO (%)
1	5	95
2	50	50
3	95	5

7.2. Infrastructure

The infrastructure employed in the experiments is distributed in a tree structure as in Fig. 4 whose parameters are presented in Table 2.

In the experiments the topology has a depth of 3, see Fig. 4. This topology yields a set of 9 DCs and 14 links. Furthermore, the demand for each application originates at the tree's leaf nodes and is propagated to the node, which hosts that particular application. This scale is sufficient to reveal the dynamics of the system. The depth is persisted throughout all experiments.

Let x be the resources consumed by an application. The operational cost is proportional to $\frac{x}{1.33}$ for a large DC, whereas it will be proportional to $\frac{x}{1.2}$ and x for medium and small DCs respectively.

7.3. Application types

The application model described in Section 6.1.3 is used to classify the applications based on their heterogeneous resource requirements as shown in Table 3. For example a compute intensive application's request (of Type 1 in Table 3) spends 5% of it execution time using compute resources and 95% time using I/O resources.

7.4. Placement algorithms

In the experiments, $\mu = 1$ and $\tilde{x}$ in Eq. (14) is set to 0.7. The execution cost and overload cost for the reference simulation system is illustrated in Fig. 5. The meeting point of the curves in Fig. 5 determines the dynamics for the cost for running applications in the system.

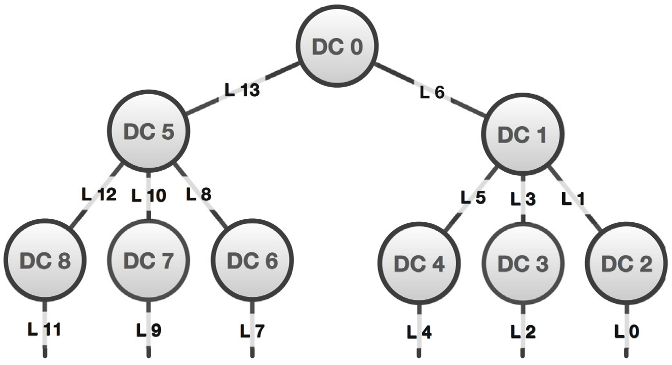


Fig. 4. Experiment infrastructure topology.

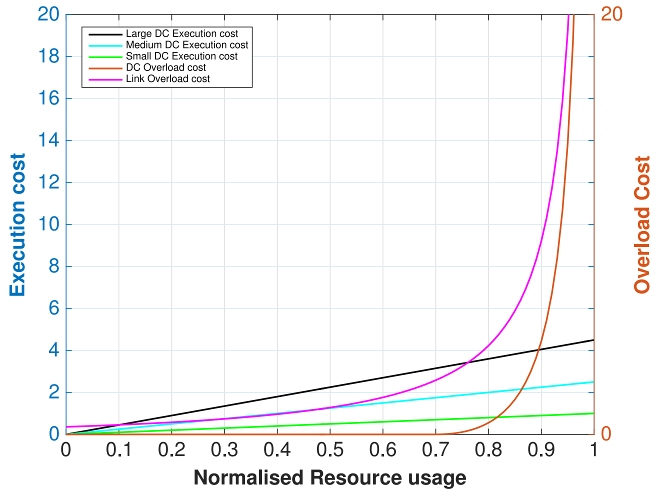


Fig. 5. Overload function vs. Resource Execution cost of large and small DC and link.

8. Results

This section presents the results of the experiments specified in Section 7. The section begins with presenting the resulting cost for each placement algorithm in Section 8.1. Sections 8.2 and 8.3 contrasts the mean RTT experienced by all applications and the mean utilisation level of each DC as a result of each placement policy, respectively.

8.1. Cost

Observing the system's aggregate cost as defined in Eq. (12) reveals how well the set of placement algorithms are able to meet the system's management objectives, at each point in time. The cost time series of the *Mobile users* scenario is presented in Fig. 6, followed by the *Campus* and *Sporting event* represented by Figs. 7 and 8 respectively.

Starting with the cost-minimising methods, Fig. 6 illustrates that for the *Mobile users* workload case, the continuous globally optimal method achieves an average 25% lower cost than when applications only are statically placed in a globally optimal manner. Additionally, the contrast between the static and the continuous approaches reveals that no matter how deliberately you place the applications as they are admitted, there is still a need to continuously re-evaluate where they are located in the infrastructure to successfully accommodate user demand mobility.

The stochastic nature of the *Mobile users* workload scenario makes it particularly challenging to manage. In such a scenario, the algorithm has no a priori knowledge of the location of the demand and therefore attempts to move the application at the rate of change of the application's demand. Furthermore, the workload in the Mobile Users scenario is transient, and thus, the volume and constellation of demand will never return to the same quantity and location. Consequently, the cost incurred by an application placed permanently in a DC, optimally or otherwise, is not likely to be either equal to or greater than the cost achieved at initial placement. As illustrated by Fig. 6, on average, when applications are randomly statically placed, the cost is 5% higher than when employing the optimal static placement scheme, but comes at a much lower computational cost.

The *Campus* and *Sporting event* workload scenarios presented in Figs. 7 and 8, respectively, are more structured since changes follow a less uncertain trajectory. In these scenarios, with a priori knowledge of workload characteristics, the random placement scheme is particularly ineffective. Therefore, the results for the random placement scheme are not included as it incurs a cost way beyond the scale of the globally and locally optimal solutions, which are the primary methods evaluated in this paper. Because of a priori knowledge, the static and continuous solutions perform nearly identically, with only a slight advantage to the continuous solution.

As illustrated by Fig. 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 Figs. 7 and 8 where the workloads are less spatially and temporally complex. In the *Campus* and *Sporting event* workloads, the cost difference between the locally optimal and globally optimal is on average $<0.5\%$.

Furthermore, the experiments also show that the search depth has little impact on the system's ability to minimise costs. A search depth of less than 4, the maximum depth of the network, introduces an occasional lag when contrasted with the optimal that is recovered in the next iteration.

8.2. RTT

Although none of the applications in this scenario have SLA constraints dictating RTT, it is worthwhile to explore the effect the cost minimisation effort has on the mean RTT across all applications. The effort of minimising the aggregate cost and managing each application's RTT are not entirely mutually exclusive. For example, minimising the cost in the manner presented in this paper, applications that are I/O intensive will tend to converge towards the origin of its demand in an attempt to reduce network usage. Moving closer to the origin of its demand also potentially reduces overall contention and thereby also latency.

Fig. 9 reveals in the *Mobile users* scenario, that applications on average experience a 35% lower RTT when applications'

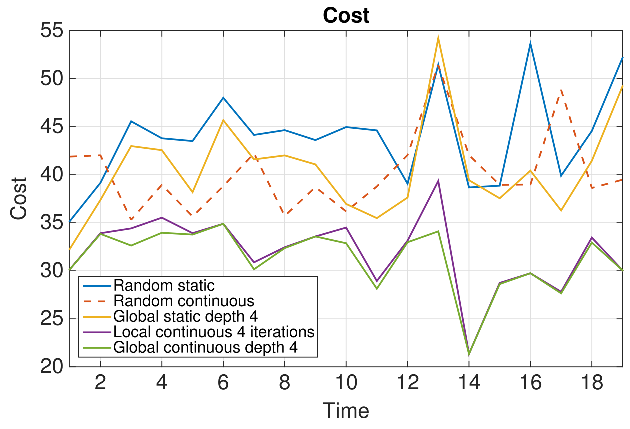


Fig. 6. Cost time-series for all placement methods for the *Mobile users* workload.

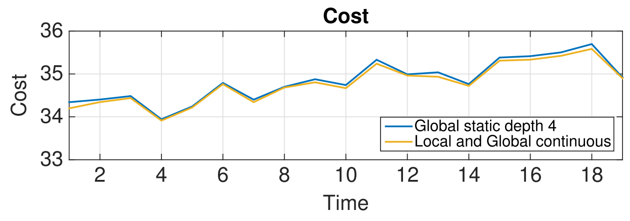


Fig. 7. Cost time-series for all placement methods for the *Campus* workload.

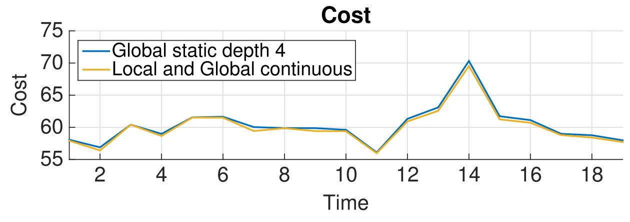


Fig. 8. Cost time-series for all placement methods for the *Sporting event* workload.

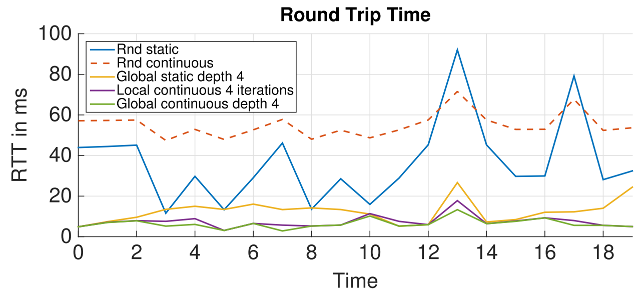


Fig. 9. RTT time-series for all placement methods for the *Mobile users* workload.

placements are continuously evaluated than when they are statically placed on admission. Intuitively, placing the applications at random results in a significantly higher RTT for each application on average as the mean network distance tends to be higher between the application instance and its demand. Furthermore, in the *Campus* workload case, as the geographic discrepancy between the application instance and its demand increases, RTT increases 10 fold across the time frame of the scenario, see Fig. 10. The uniformity of demand across all applications and the stationarity of the centre of demand leave 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 applications are able to relocate to accommodate the spatial surge in demand and mitigating some of the incurred latency, see Fig. 11.

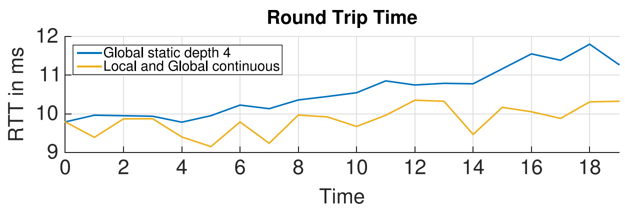


Fig. 10. RTT time-series for all placement methods for the *Campus* workload.

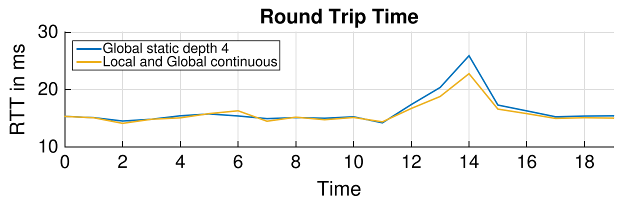


Fig. 11. RTT time-series for all placement methods for the *Sporting event* workload.

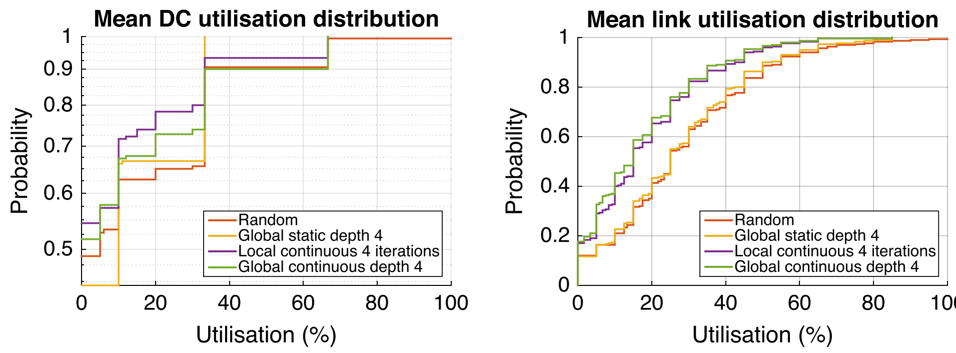
8.3. Resource utilisation

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. Attention is now directed at how well the methods mitigate skewed resource utilisation. The algorithms attempt to avoid overloading individual components in the system by progressively penalising high utilisation levels through the penalty component in the objective function, defined in Eq. (14).

Figs. 12–14 reveal that network utilisation is reduced when applications' placements are continuously evaluated. The jagged shape of the DC utilisation CDF exhibited in Fig. 12(a) can be explained by the stochastic discrete movement of the demand, from one branch of the tree to another. The stochastic nature of the *Mobile users* workload can also be attributed to the more gradual ascent of the link utilisation CDF. Although it exhibits discrete levels, the depth of the network and the number of links increase the granularity of the utilisation levels.

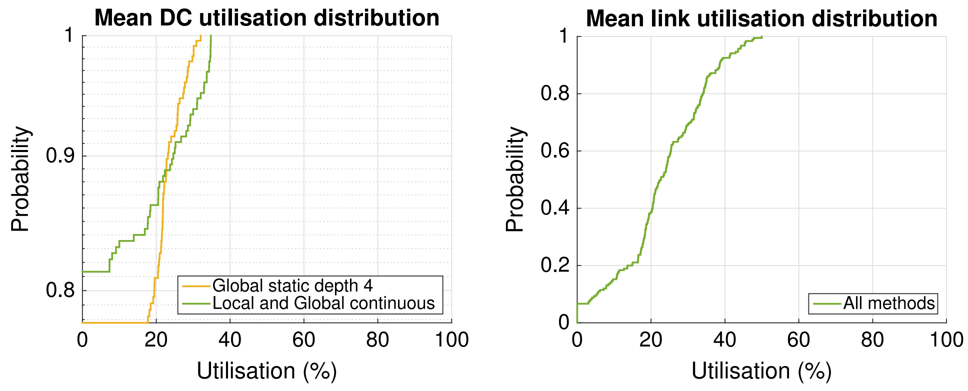
Looking closer at the outcome when employing continuous re-evaluation, the globally optimal algorithm achieves lower mean utilisation levels than the locally optimal algorithm. Fig. 12(b) shows that the mean link resource utilisation in the Mobile user case is reduced by 5% in favour of the optimal search over the local search method. On the other hand, mean DC utilisation is increased by 5%. Fig. 12(b) shows that the continuous methods persistently achieve lower utilisation levels but with a similar gradient. This property can be attributed to the fairly persistent mean spatial offset to the optimal placement. A larger network would thus exhibit a greater separation. Furthermore, the 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.

Having observed the properties of the *Mobile user* scenario, attention is now directed to the structured workloads of *Campus* and *Sporting event*. Figs. 13(b) and 14(b) illustrate that no algorithm has a mean link utilisation advantage. The results are thus represented as one graph. Again, this can be attributed to the structured nature of the workload where demand is confined in space. Additionally, the steep ascent utilisation in Fig. 14(b) is a consequence of the high concentration of demand in the *Sporting event* scenario, which confines the application to a small group of DCs. This effect also manifests itself as a higher mean utilisation level for the *Sporting event* scenario. Nevertheless, the mean DC utilisation level is more clustered and is confined to a narrow range when using the continuous globally and locally optimal algorithms



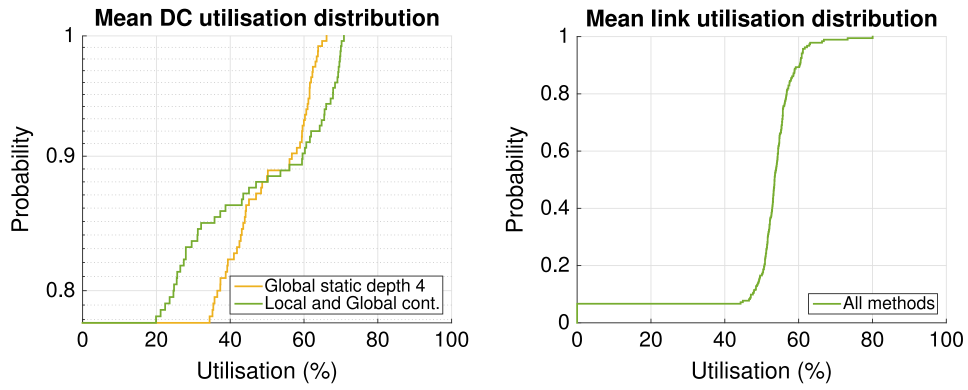
(a) DC utilisation. (b) Link utilisation.

Fig. 12. Data Centre and Link utilisation for the *Mobile users* workload.



(a) DC utilisation. (b) Link utilisation.

Fig. 13. Data Centre and Link utilisation for the *Campus* workload.



(a) DC utilisation. (b) Link utilisation.

Fig. 14. Data Centre and Link utilisation for the *Sporting event* workload.

over the static globally optimal placement scheme, see Figs. 13(a) and 14(a). In either case, the utilisation levels are kept at a desirable level.

9. Related work

In this section, related works are reviewed, and how the results from the literature can be employed in MCN research. Remaining challenges not covered by the current literature are also covered. There exists an extensive body of work in the field of content and service placement in distributed compute and content delivery systems. Existing research results address many of the challenges facing an MCN.

9.1. Replica placement

There have been numerous efforts at developing algorithms for replicas in a network of computers [42]. In general, their objectives are to either optimise the applications' performance in an existing infrastructure, to reduce the computational complexity of the decision, or minimise the infrastructure cost while meeting the applications' SLAs. The suboptimal algorithm proposed in [43] 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, [44] proposes a topology-aware replica placement. The two algorithms achieve near optimal placement of replicas, but do not address the mobility of the services' users and thus the continuous evaluation of the services' placement in the network.

9.2. 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 [45] and distributed [46] 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 CDN's objectives. Due to the nature of CDN infrastructures, the aforementioned methods do not take into account user mobility, application/content component affinity and execution, and finite heterogeneous resources.

9.3. Inter-and-Intra data centre VM-placement

Research in Intra-DC VM placement and continuous placement evaluation has yielded methods [47,26,48,49] 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 Physical Machine mappings. As the placement domain is either within a DC or in a set of DCs [50] the research fails to take into account end-user locality and rapid spatial and temporal demand changes in demand.

The authors of [51] 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 Round-Trip Delay time (RTD). The work highlights the inherit challenges of what to migrate, duplicate, and replicate, and how to evaluate the actions' performance return. The work however does 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 [52], 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.

The authors of [53] designed an algorithm for packing VMs across multiple heterogeneous DCs that hedges for dynamic demand. Similarly, the authors of [54] designed an application-centric, greedy algorithm with the objective to locate the application where it will experience the least amount of network congestion. The algorithms presented in [55,56] follow a similar approach but include the objective to maximise the VMs fault/failure tolerance. Nevertheless, neither one of these three approaches takes into account proximity to end-users or operational cost of the infrastructure.

10. Conclusions

One of the foremost challenges in an MCN paradigm is how to manage the highly heterogeneous and distributed resources in a complex system. This paper contributes with a system model for the MCN and an objective function to minimise the global system cost as a means to manage the compute and network resources in an MCN. Based on this model, a globally optimal placement of static and mobile applications was designed.

Further, a locally optimal placement scheme with a fraction of the computational cost was designed. Additionally, a set of near-optimal and intuitive methods is used to contrast the performance of the presented algorithms. Also, based on the presented model, a simulation environment was developed on which the algorithms are evaluated using a set of challenging MCN workloads.

The results reveal that when user demand is highly mobile and stochastic, significant resource utilisation and cost gains can be made when one employs any method that attempts to map the location where the application is executed with the location of that application's demand. Furthermore, the experiments also reveal that the globally optimal and locally optimal schemes can achieve near equal performance with workloads that have a spatially uniform distribution of demand. A difference in performance between the algorithms was uncovered when subjecting the system to a workload with a spatially non-uniform demand. Here, the globally optimal approach outperforms the locally optimal. Nevertheless, with a transient workload, system cost and resource utilisation are with either algorithm non-divergent.

For future work, it will be beneficial to extend the experiments to include application component affinity to model the performance of for example multi component applications where each component will also be permitted to scale horizontally in the network to meet a spatially distributed demand. From an algorithm perspective, extending the current work to include a controller that regulates the rate of re-evaluation as to further minimise the computational complexity. Additionally, having revealed the upper bound of the optimal solution, future experiments can now be confidently carried out with tractable near-optimal solutions at a later scale. Larger scale experiments will increase the resolution of the experiment and amplify the dynamics of the system and the potency of the algorithms. The presented algorithms can also be applied to other problems with the objective to continuously place time-variant spatially- and quantitatively-heterogeneous entities in cost- and capacity-heterogeneous vessels.

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] L.A. Barroso, U. Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2009.
[2] Z. Sanaei, S. Abolfazli, A. Gani, R. Buyya, Heterogeneity in mobile cloud computing: Taxonomy and open challenges, IEEE Commun. Surv. Tutorials 16 (1) (2014) 369–392. http://dx.doi.org/10.1109/SURV.2013.050113.00090.

[3] D. Evans, The Internet of everything: How more relevant and valuable connections will change the world, 2012, pp. 1–9.

[4] R. Shea, J. Liu, E.-H. Ngai, Y. Cui, Cloud gaming: architecture and performance, IEEE Netw. 27 (4) (2013) 16–21. http://dx.doi.org/10.1109/MNET.2013.6574660.

[5] G. Karagiannis, A. Jamakovic, A. Edmonds, C. Parada, T. Metsch, D. Pichon, M. Corici, S. Ruffino, A. Gomes, P.S. Crosta, T.M. Bohnert, Mobile cloud networking: Virtualisation of cellular networks, in: 2014 21st International Conference on Telecommunications, ICT, 2014, pp. 410–415. http://dx.doi.org/10.1109/ICT.2014.6845149.

[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, Vol. 53, ACM, 2010, pp. 50–58.

[7] P. Bosch, A. Duminuco, F. Pianese, T. Wood, Telco clouds and virtual telco: Consolidation, convergence, and beyond, in: 2011 IFIP/IEEE International Symposium on Integrated Network Management, IM, 2011, pp. 982–988. http://dx.doi.org/10.1109/INM.2011.5990511.

[8] E. Ahmed, A. Gani, M.K. Khan, R. Buyya, S.U. Khan, Seamless Application Execution in Mobile Cloud Computing: Motivation, Taxonomy, and Open Challenges, Vol. 52, Elsevier, 2015, pp. 154–172.

[9] N. Fernando, S.W. Loke, W. Rahayu, Mobile Cloud Computing: A Survey, Vol. 29, Elsevier, 2013, pp. 84–106.

[10] A. Ahmed, E. Ahmed, A survey on mobile edge computing, in: Accepted to the 10th IEEE INternational Conference on Intelligent Systems and Control, IEEE, 2016.

[11] Y. Saleem, F. Salim, M.H. Rehmani, Resource management in mobile sink based wireless sensor networks through cloud computing, in: Resource Management in Mobile Computing Environments, Springer, 2014, pp. 439–459.

[12] S. Ghafoor, M.H. Rehmani, S. Cho, S.-H. Park, An Efficient Trajectory Design for Mobile Sink in a Wireless Sensor Network, Vol. 40, Elsevier, 2014, pp. 2089–2100.

[13] E. Ahmed, A. Gani, M. Sookhak, S.H. Ab Hamid, F. Xia, Application Optimization in Mobile Cloud Computing: Motivation, Taxonomies, and Open Challenges, Vol. 52, Elsevier, 2015, pp. 52–68.

[14] T. Liu, F. Chen, Y. Ma, Y. Xie, An Energy-efficient Task Scheduling for Mobile Devices Based on Cloud Assistant, Vol. 61, Elsevier, 2016, pp. 1–12.

[15] S.W. Loke, Supporting Ubiquitous Sensor-cloudlets and Context-cloudlets: Programming Compositions of Context-aware Systems for Mobile Users, Vol. 28, Elsevier, 2012, pp. 619–632.

[16] Y.-S. Chang, C.-T. Fan, W.-T. Lo, W.-C. Hung, S.-M. Yuan, Mobile Cloud-based Depression Diagnosis Using an Ontology and a Bayesian Network, Vol. 43, Elsevier, 2015, pp. 87–98.

[17] M. Rehmani, Cognitive Radio Sensor Networks: Applications, Architectures, and Challenges: Applications, Architectures, and Challenges, Advances in Wireless Technologies and Telecommunication, IGI Global, 2014.

[18] M.H. Rehmani, A.-S. K. Pathan, Emerging Communication Technologies Based on Wireless Sensor Networks, CRC Press, Taylor and Francis Group, USA, 2015.

[19] 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, F. Galan, The reservoir model and architecture for open federated cloud computing, IBM J. Res. Dev. 53 (4) (2009) 4:1–4:11. http://dx.doi.org/10.1147/JRD.2009.5429058.

[20] U. Shaukat, E. Ahmed, Z. Anwar, F. Xia, Cloudlet Deployment in Local Wireless Networks: Motivation, Architectures, Applications, and Open Challenges, Vol. 62, Elsevier, 2016, pp. 18–40.

[21] C. Adam, R. Stadler, Service middleware for self-managing large-scale systems, IEEE Trans. Netw. Serv. Manag. 4 (3) (2007) 50–64. http://dx.doi.org/10.1109/TNSM.2007.021103.

[22] Y. Zaki, L. Zhao, C. Goerg, A. Timm-Giel, LTE wireless virtualization and spectrum management, in: Wireless and Mobile Networking Conference, WMNC, 2010 Third Joint IFIP, 2010, pp. 1–6. http://dx.doi.org/10.1109/WMNC.2010.5678740.

[23] S. Singh, P. Singh, Key Concepts and Network Architecture for 5g Mobile Technology, Vol. 1, 2012, pp. 165–170.

[24] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, A. Akella, On the Treeness of Internet Latency and Bandwidth, Vol. 37, ACM, New York, NY, USA, 2009, pp. 61–72. http://dx.doi.org/10.1145/2492101.1555357.

[25] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, 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, 2011, pp. 72–79. http://dx.doi.org/10.1109/SCC.2011.28.

[26] X. Meng, V. Pappas, L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, in: INFOCOM, 2010 Proceedings IEEE, 2010, pp. 1–9. http://dx.doi.org/10.1109/INFCOM.2010.5461930.

[27] L.A. Barroso, J. Clidaras, U. Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines, Vol. 8, Morgan & Claypool Publishers, 2013, pp. 1–154.

[28] A. ElNashar, M. El-saidny, M. Sherif, Design, Deployment and Performance of 4G-LTE Networks: A Practical Approach, first ed., Wiley, 2014.

[29] D. Schwab, 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.

[30] G. Zipf, Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology, Martino Publishing, 2012.

[31] P. Bodik, A. Fox, M.J. Franklin, M.I. Jordan, D.A. Patterson, Characterizing, modeling, and generating workload spikes for stateful services, in: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC'10, ACM, New York, NY, USA, 2010, pp. 241–252. http://dx.doi.org/10.1145/1807128.1807166.

[32] A. Mehta, J. Durango, J. Tordsson, E. Elmroth, Online spike detection in cloud workloads, in: 2015 IEEE International Conference on Cloud Engineering, IC2E, 2015, pp. 446–451. http://dx.doi.org/10.1109/IC2E.2015.50.

[33] M. Arlitt, T. Jin, A workload characterization study of the 1998 world cup web site, IEEE Netw. 14 (3) (2000) 30–37. http://dx.doi.org/10.1109/65.844498.

[34] T. Wood, P.J. Shenoy, A. Venkataramani, 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, 17–17.

[35] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, N. Sharma, Towards autonomic workload provisioning for enterprise grids and clouds, in: 2009 10th IEEE/ACM International Conference on Grid Computing, 2009, pp. 50–57. http://dx.doi.org/10.1109/GRID.2009.5353066.

[36] X. Qin, H. Jiang, A. Manzanares, X. Ruan, S. Yin, Dynamic Load Balancing for I/O-intensive Applications on Clusters, Vol. 5, ACM, 2009, p. 9.

[37] S. Kundu, R. Rangaswami, K. Dutta, 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.

[38] D. Carrera, M. Steinder, I. Whalley, J. Torres, E. Ayguade, Autonomic placement of mixed batch and transactional workloads, IEEE Trans. Parallel Distrib. Syst. 23 (2) (2012) 219–231. http://dx.doi.org/10.1109/TPDS.2011.129.

[39] T.R. Henderson, M. Lacage, G.F. Riley, C. Dowell, J. Kopena, Network Simulations with the ns-3 Simulator, Vol. 14, 2008.

[40] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Vol. 41, Wiley Online Library, 2011, pp. 23–50.

[41] N. Matloff, Introduction to Discrete-event Simulation and the Simpy Language, Vol. 2, 2008, p. 2009.

[42] M. Karlsson, 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.

[43] M. Szymaniak, G. Pierre, M. van Steen, Latency-driven replica placement, in: Proceedings of the 2005 Symposium on Applications and the Internet, 2005., 2005, pp. 399–405. http://dx.doi.org/10.1109/SAINT.2005.37.

[44] P. Radoslavov, R. Govindan, D. Estrin, Topology-informed Internet Replica Placement, Vol. 25, 2002, pp. 384–392.

[45] L. Qiu, V. Padmanabhan, G. Voelker, On the placement of web server replicas, in: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, vol. 3, IEEE, 2001, pp. 1587–1596. http://dx.doi.org/10.1109/INFCOM.2001.916655.

[46] S. Zaman, D. Grosu, A distributed algorithm for the replica placement problem, IEEE Trans. Parallel Distrib. Syst. 22 (9) (2011) 1455–1468. http://dx.doi.org/10.1109/TPDS.2011.27.

[47] P. Svärd, W. Li, E. Wadbro, J. Tordsson, E. Elmroth, Continuous Datacenter Consolidation, no. 2014:08 in Report/UMINF, Umeåuniversitet, 2014, p. 12.

[48] F.P. Tso, G. Hamilton, K. Oikonomou, D. Pezaros, Implementing scalable, network-aware virtual machine migration for cloud data centers, in: 2013 IEEE Sixth International Conference on Cloud Computing, CLOUD, 2013, pp. 557–564. http://dx.doi.org/10.1109/CLOUD.2013.82.

[49] D. Espling, L. Larsson, W. Li, J. Tordsson, E. Elmroth, Modeling and placement of structured cloud services, 2014.

[50] B. Kantarci, L. Foschini, A. Corradi, H. Mouftah, Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems, in: Globecom Workshops, GC Wkshps, 2012, pp. 708–713. http://dx.doi.org/10.1109/GLOCOMW.2012.6477661.

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

[52] N. Laoutaris, M. Sirivianos, X. Yang, P. Rodriguez, Inter-datacenter Bulk Transfers with Netstitcher, Vol. 41, ACM, 2011, pp. 74–85.

[53] Y. Guo, A.L. Stolyar, A. Walid, Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud, in: INFOCOM, 2013 Proceedings IEEE, IEEE, 2013, pp. 620–628.

[54] M. Mihailescu, S. Sharify, C. Amza, Optimized application placement for network congestion and failure resiliency in clouds, in: 2015 IEEE 4th International Conference on Cloud Networking, CloudNet, 2015, pp. 7–13. http://dx.doi.org/10.1109/CloudNet.2015.7335272.

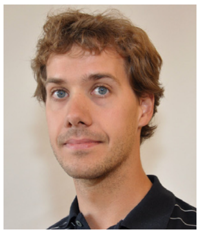
[55] M. Mihailescu, S. Sharify, C. Amza, Optimized application placement for network congestion and failure resiliency in clouds, in: 2015 IEEE 4th International Conference on Cloud Networking, CloudNet, IEEE, 2015, pp. 7–13.

[56] B. Spinnewyn, B. Braem, S. Latre, Fault-tolerant application placement in heterogeneous cloud environments, in: 2015 11th International Conference on Network and Service Management, CNSM, IEEE, 2015, pp. 192–200.

William Tärneberg is a Ph.D. candidate at the Department of Electrical and Information Technology at Lund University in Sweden. William received his M.Sc. in Electrical Engineering from Lund University in 2010. Prior to beginning his Ph.D. studies he worked as a Development and Research Engineer for Sony. His research interests include distributed systems and algorithms, machine learning, traffic shaping, and network simulation.

Amardeep Mehta is a Ph.D. candidate at the Department of Computing Science, Umeå University, Sweden. Amardeep received his M.Sc. in Computer Science from Uppsala University, Sweden. His research interests are workload analysis and distributed systems. Amardeep has also worked as a Software Developer at Ericsson in Stockholm, Sweden.

Eddie Wadbro received his M.Sc. degree in Mathematics at Lund University in 2004 and his Licentiate and Ph.D. degrees in Scientific Computing at Uppsala University in 2006 and 2009, respectively. He works as an Assistant Professor at the Department of Computing Science, Umeå University. His research interests concern development and analysis of efficient numerical methods, primarily within the fields of design optimisation and inverse problems.


Johan Tordsson is Associate Professor at the Department of Computing Science, Umeå University, from where he also received his Ph.D. in 2009. After a period as visiting postdoc researcher at Universidad Complutense de Madrid, he worked for several years in the RESERVOIR, VISION Cloud, and OPTIMIS European projects, in the latter as Lead Architect and Scientific Coordinator. Tordsson's research interests include autonomic management problems for clouds and data centres as well as enabling technologies such as virtualization.

Johan Eker is a Principal Researcher at Ericsson, Sweden. He received his Ph.D. in automatic control from Lund University in 1999. After a brief stint in industry he joined the Ptolemy group at UC Berkeley in the US in 2001 working on the CAL Actor Language, which later was turned into a global ISO standard by the MPEG organisation. He has been with Ericsson Research since 2003 working on embedded software for mobile phones, including operating systems, programming languages, compilers, and distributed systems. During the period 2008–2011 he acted as the coordinator of the European FP-7 research project ACTORS. Since February 2013 he is also adjoint Associate Professor at Lund University.

Maria Kihl is Professor in Internetworked Systems at the Department of Electrical and Information Technology, Lund University, Sweden. Her work focuses on performance modelling, analysis, and control of distributed Internet-based systems, currently Cloud systems and media distribution architectures.

Erik Elmroth is Professor in Computing Science at Umeå University. He has been Head and Deputy Head of the Department of Computing Science for ten years. He has established the Umeå University research on distributed systems, a addressing virtual computing infrastructures (Grid and Cloud computing), see http://www.cloudresearch.org. The research is focused on methods, algorithms, architectures, and software design principles for large scale distributed environments. The current research extends on Elmroth's broad background in scientific and high-performance computing and extensive experience from organising supercomputing infrastructures. International experiences include a year at NERSC, Lawrence Berkeley National Laboratory, University of California, Berkeley, and one semester at the Massachusetts Institute of Technology (MIT), Cambridge, MA.