An Energy-Efficient Multi-Cloud Service Broker for Green Cloud Computing Environment

By Bandar Aldawsari

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

October 2017

Abstract

The heavy demands on cloud computing resources have led to a substantial growth in energy consumption of the data transferred between cloud computing parties (i.e., providers, datacentres, users, and services) and in datacentre's services due to the increasing loads on these services. From one hand, routing and transferring large amounts of data into a datacentre located far from the user's geographical location consume more energy than just processing and storing the same data on the cloud datacentre. On the other hand, when a cloud user submits a job (in the form of a set of functional and non-functional requirements) to a cloud service provider (aka, datacentre) via a cloud services broker; the broker becomes responsible to find the best-fit service to the user request based mainly on the user's requirements and Quality of Service (QoS) (i.e., response time, latency).

Hence, it becomes a high necessity to locate the lowest energy consumption route between the user and the designated datacentre; and the minimum possible number of most energy efficient services that satisfy the user request. In fact, finding the most energy-efficient route to the datacentre, and most energy efficient service(s) to the user are the biggest challenges of multi-cloud broker's environment.

This thesis presents and evaluates a novel multi-cloud broker solution that contains three innovative models and their associated algorithms. The first one is aimed at finding the most energy efficient route, among multiple possible routes, between the user and cloud datacentre. The second model is to find and provide the lowest possible number of most energy efficient services in order to minimise data exchange based on a bin-packing approach. The third model creates an energy-aware composition plan by integrating the most energy efficient services, in order to fulfil user requirements. The results demonstrated a favourable performance of these models in terms of selecting the most energy efficient route and reaching the least possible number of services for an optimum and energy efficient composition.

Introduction 9

1.1 Introduction 10
1.1.2 Motivation: energy consumption reduction 12
1.1.3 Research Hypothesis 13
1.1.4 Research Problem 13
1.1.5 Novelty of the thesis 15
1.1.6 Research Scope 16
1.1.7 Aim and Objectives 16
1.1.8 Methodology 18
1.1.9 Thesis Structure 18

Cloud Computing 20

2.1. Introduction 21

2.2. Cloud Computing 21
2.2.1. Deployment models of cloud computing 23
2.2.2. The main players in cloud computing 25

2.3. Cloud Computing Virtualization 26
2.3.1. Virtualization Forms 27
2.3.2. Categories of server virtualisation 28
2.3.2.1. Hypervisor based virtualization: 28
2.3.2.2. Container based virtualization: 29

2.4. Cloud Services 30
2.4.1. Cloud service models 30
2.4.1.1. Infrastructure as a Service (IaaS): 30
2.4.1.2. Platform as a Service (PaaS): 31
2.4.1.3. Software as a Service (SaaS): 31
2.4.2. New hybrid service models 32

2.5. Cloud Service Brokerage 33
2.5.1. Cloud Service Broker Architectures 35
2.5.2. Resource Management in Cloud Broker 35

2.6. Summary 37

Network Routing and Energy Efficiency in Cloud Computing 38

3.1. Introduction 39
3.1.1. Cloud broker problem 39
3.1.1.1. Load Balancing Algorithms 42
3.1.2. Segment Routing 44
3.1.2.1 Traffic engineering 47
3.1.3. Routing algorithm to balance energy consumption 49

3.2. Energy efficiency in cloud computing 52
3.2.1. Energy Consumption In Cloud Datacentres 52
3.2.1.1 The main factors leading to waste-to-energy 53
3.2.1.2 Power saving strategies in Cloud 54
3.2.2 Energy Efficient Cloud Resources Allocation 57
3.2.2.1 Resources Allocation 57
3.2.2.2 Cloud Resources Allocation 58
3.2.2.3 On-demand resource allocation vs advanced resource reservation 60
3.2.2.4 Static vs dynamic Cloud resources allocation 62

3.2.3 Energy efficient service composition **65**
3.2.3.1. Bin-packing approach 68
3.3. Discussion and Requirements **69**
3.3.1. Requirements **71**
3.5. Summary **72**
An Energy Efficient Routing Algorithm **73**
4.1. Introduction **74**
4.2 Energy Efficient Routing **74**
4.2.1 Basics and Rules **76**
4.2.2 Modelling power consumption of the network **77**
4.2.3 Modelling user connectivity to data centre **78**
4.2.4 Formal analysis of network topology **79**
4.2.5 Energy required for transportation **81**
4.2.6 Time required for transportation **83**
4.2.7 Energy and time required for computation **83**
4.3. Implementation **83**
4.3.1. Linear programming formulation **83**
4.3.2. Goal programming formulation **86**
4.3.3. Dynamic programming approach **87**
4.4. Evaluation **88**
4.4.1. Physical topology **89**
4.4.2. Energy evaluation model and results **91**
4.5. Summary **94**
Bin-Packing Based Energy-Efficient Service Provision **95**
5.1. Introduction **96**
5.1.1. Service Composition Energy Consumption **96**
5.2. The System Model **99**
5.2.1. Formal datacentre-broker model **99**
5.2.2. Formal datacentre-broker model **104**
5.3. Implementation **105**
5.3.1. Algorithmic Design **105**
5.4. Evaluation **112**
5.4.1 Experimental Settings **112**
5.4.2 Experimental results **113**
5.5. Summary **116**
Energy-Aware Service Composition Algorithm for Multiple Cloud **117**
6.1. Introduction **118**
6.1.1. Service compositions **118**
6.2. Model design **122**
6.2.1. User-broker model **122**
6.2.2. Datacentre-broker model **124**
6.3. Implementation **126**
6.3.1. Optimal service composition plan **126**

6.4 Evaluation 132
6.4.1. Experimental settings 132
6.4.2 Results and analysis 135
6.5. Summary 140
Conclusion and Future Work 141
6.1. Conclusion 142
6.2. Contributions to knowledge 143
6.3. Future work 145
7. References 147
Appendix 1 155
Appendix 2 156

List of Figures

Figure 2.1 Cloud Computing concept (F. Liu et al. 2011) 22
Figure 2.2 number of isolated virtual servers to run on one single actual server (editor 2011) 27
Figure 2.3. Container based virtualization vs hypervisor based virtualization (Travostino et al. 2006) 28
Figure 2.4 three types of classic cloud service models (Kubernetes 2017) 30
Figure 2.5 cloud service brokerage roles to manage the broker 33
Figure 3.6 Resource Allocation (Nair and Porwal 2010) 59
Figure 4.7 cloud elements, that contribute to the total energy consumption. 75
Figure 4.8 Network structure (user connectivity to data centre) 78
Figure 4.9 Hierarchal topology of an Italian ISP 89
Figure 4.10 Path lengths of each route. 92
Figure 4.11 Total energy of each route. 93
Figure 4.12 Average energy consumption per node of each route. 94
Figure 5.13 Conceptual representation of the proposed approach. 103
Figure 5.14 Four main steps broker 107
Figure 5.15 Running time and energy consumption to find requested service 116
Figure 6.16: Five separate web services. 120
Figure 6.17 : BPMN composition 120
Figure 6.18 A conceptual representation of the proposed approach. 124
Figure 6.19 % reduction in the number of examined atomic services. 138
Figure 6.20 The average number of examined services. 138
Figure 6.21 Number of combined clouds in multi-cloud based service composition plan π. 139
Figure 6.22 Running Time to achieve service composition plan π. 140

List of Tables

Table 3.1 Proposed Broker Requirements 72
Table Algorithm 4.1.: Energy Efficiency Algorithm..... **Error! Bookmark not defined.**
Table 4.3: Route A network components 90
Table 4.4 Route B network components..... 91
Table 4.5: Route C network components..... 91
Table 4.6 Route D network components..... 91
Table 5.7 Multiple-cloud providers and Services 104
Table 5.8 Multiple-cloud providers and Services sorted by energy consumption..... 105
Table Algorithm 5.1: Ordering the Cloud Providers in a descending order based on Total Energy Consumption 108
Table Algorithm 5.2: Finding an atomic service that matches the user request 109
Table Algorithm 5.3: Finding a predefined optimal composition plan $\pi B'$ from a single provider 110
Table Algorithm 5.4**:** Creating an optimal services composition from multiple providers 111
Table 5.13 Cloud provider's composition set per MCP. 112
Table 5.14 Number of services per composition. 113
Table 5.15: CPs and energy consumption per MCPs, before listing in an ascending order. 113
Table 5.16: CPs and energy consumption per MCPs, after listing in an ascending order... 113
Table 5.17: CPs and number of () composition plans per MCPs..... 114
Table 6.18 Multiple-cloud providers and services..... 127
Table Algorithm 6.1 discovering a predefined optimal composition plan $\pi B'$ from a single provider. 130
Table algorithm 6.2 The creation of a dynamic optimal composition plan $\pi B'$ from multiple providers. 131
Table 6.21 Cloud providers composition set per multiple-cloud providers environment. .. 134
Table 6.22 Number of services per composition plans..... 134
Table 6.23 CPs and number of (π) composition plans per MCPs..... 135
Table 6.24 CPs and number of (π) composition plans per MCPs. 136
Table Summary of notations used..... 156

ACKNOWLEDGMENTS

This thesis is the story of a journey throughout which I received help from many people, some of them helped me from outside and others gave me unforgettable internal motivation. I would like to thank my parents and my family for their love, patience and support at all times. I would like to thank my supervisor, Dr. Thar Baker for his intellectual guidance and continuous encouragement, which ensured the successful completion of this thesis. He always gave me freedom to think broadly and deeply into my research. The regular meetings conducted by him made me work more regularly and systematically. His endless amount of energy is really an inspiration for me to do better research. Also, I would like to thank Dr David England for his guidance and encouragement during my study. I would like also to thank Dr Martin Randles for his support.

Introduction

1.1 Introduction

Thanks to its on-demand utility pricing model, cloud computing has grown considerably to support online services both for businesses and for individuals. The business model is deemed simple and powerful: providers, offer services, end-users find the service they want, and when they have subscribed, high-speed network connection is put in place between them to enable them to use the selected services. As an advanced technology, it has revolutionised the Information and Communications Technology (ICT) industry. In other words, cloud computing has changed the way that services are offered through the Internet. It provides computing resources such as hardware, application development platforms and computer applications available as services over the Internet. These services are commonly known, respectively, as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). It represents a shift in the geography of computation, where the cloud resources' physical location is not a barrier whatsoever for the users and providers. Therefore, the cloud computing resources can be accessed at anytime from anywhere in the world via the users' machines that are connected to the Internet (Larumbe and Sansò 2012). Hence, on the one hand the users do not need to worry about where the resources/services are based and/or how they can be accessed and used. And on the other hand, cloud providers can offer their services/resources to anyone on the globe. Furthermore, cloud computing provides various benefits in terms of scalability, cost, maintenance and performance compared to the traditional computing (Baliga et al. 2011).

Based on the above, the heavy demands on cloud computing resources has led to a substantial growth in the data transferred between cloud computing parties (i.e., providers, data centres, users, and services), network traffic and the energy consumed by the huge infrastructure of cloud computing, which is needed to meet the users' requests quickly and effectively. The cloud network traffic is forecasted to increase about threefold between 2013

and 2018 (Larumbe and Sansò 2012). Routing big data between the cloud computing parties requires a high bandwidth connection, which consumes larger amounts of energy (Chang Ge, Zhili Sun, and Ning Wang 2013) than just processing and storing big data on cloud data centres, and thus, producing high carbon dioxide emissions. When transferring such amounts of data into a data centre located quite far from the users' geographical location, this power consumption becomes significantly high. Generally, there are two main pillars for energy consumption of cloud computing that should be dealt with efficiently and equally to achieve the most energy-efficient cloud computing environment: (i) the amount of energy consumed at the computation and processing of each service at the data centre and (ii) the amount of energy consumed on transporting the data between the user and the cloud data centre. Hence, it has become a high necessity to locate the lowest energy consumption route between the user and the designated data centre, and services that match the user request, while making sure the users' requirements are met.

In its simplest form, one cloud provider provides all services required by a user. However, where this is not possible (i.e., more than one cloud provider is involved in producing the required service(s)) the service's components/parts must be put together with care. A user's request with a specification of the services required goes to the provider, who returns a reply to the user that contains the required services. When resource limitations or other constraints mean that a single service provider cannot meet all of the user's needs, the model becomes more complex. Typically, this will be seen as composition of a number of services and the use of the model of broker-based cloud services (Aldawsari, Baker, and England 2015), with multiple cloud service providers collaborating in a multi-cloud environment so that the user can receive the desired service outcomes. The assumption must be that the various services coordinate their activities to avoid conflict in such matters as shareable resources and order dependencies, and also to avoid delays in communication.

This presents a huge challenge due to the following points:

- To meet environmental requirements as published in the 2011 report of PBL Netherlands Environmental Assessment Agency and JRC European Commission; (Jos G.J. Olivier, Greet Janssens-Maenhout, Jeroen A.H.W. Peters 2011).
- To reduce energy consumption (Baer 2008) and lower the volume of CO_2 emissions by 15%-30% before 2020 to keep increases in global temperature below 2^oC.
- To integrate fewer services from cross-continental and scattered providers to fulfil user demand, meaning that there is less data exchanged, leading to a reduction in the carbon footprint of the multi-cloud IoT environment. Moving lots of data between services has an impact on energy.

This thesis present three new models and their related algorithms to address the above issue. In the first one, the most energy efficient route between the user and cloud datacentre will be chosen in order to achieve the full green cloud computing. The second algorithm is based on a bin-packing approach and aimed at finding and integrate the lowest possible number of services in order to minimise data exchange. The last model focuses on the creation of energy-aware composition plan in order to fulfil user requirements, by searching for and integrating the most energy efficient services.

1.1.2 Motivation: energy consumption reduction

The Energy White Paper, published on 24 February 2003, sets out the government's overall goals for the UK energy policy (Department for Transport 2003), and puts the UK on a path to cut CO_2 emission by 60% by 2050. In addition, according to an IEEE Communication Survey (Chang Ge, Zhili Sun, and Ning Wang 2013), the global electricity demand of cloud data centres was 623bn kWh in 2007, 416.2 tWh in 2015, which was significantly higher

than the UK's total consumption of about 300 tWh, and is expected to rise up to 1963.74 tWh by 2020. For instance, a data centre in North Carolina consumes as much as 100MW of power, which is equivalent to about 80,000 US homes or 250,000 EU homes (Baer 2008).

As explained above, in a typical scenario in a cloud environment, a user submits a job to a cloud service broker explaining the requirements of the needed services. The broker should then find the appropriate service provider or set of providers that serve the request. Yet, finding the most energy efficient route to a data centre and locating the best fit and the most energy efficient service, or set of services, that matches the user needs are the most challenging tasks for the multi-cloud brokers. The state-of-the-art in this domain focuses on the power consumption of the datacentre and its equipment, including server, storage and network equipment as well as new cooling and power management technologies.

Thus, this was the spark to conduct research in this area and find the best solution to solve the above-mentioned issues.

1.1.3 Research Hypothesis

The research hypothesis of this work is that, selecting the most energy efficient route to the cloud data centre alongside finding and integrating the lowest possible number of most energy efficient services from the least possible number of cloud providers can help in achieving a full green cloud computing environment, which eventually contributes to the energy saving goals by 2050.

1.1.4 Research Problem

Routing and transferring large amounts of data into a data centre located quite far from the users' geographical location can consume larger amounts of energy than just processing and storing the same data on cloud data centres. Hence, it became a high necessity to locate the lowest energy consumption route between the user and the designated data centre, while

making sure the users' requirements are met. On the other hand, when a cloud user submits a job (in the form of a set of functional and non-functional requirements) to a cloud service provider (data centre) via a cloud services broker; the broker is then responsible to find the best-fit service to the user request based on the user's requirements. Finding the most energy-efficient route to the data centre, and best-fit service, in terms of energy consumption, to the user is the multi-cloud broker's biggest challenge because:

- When a user request is submitted, all routes towards the data centre do not consume the same amount of energy. Therefore, a routing solution is needed to find the most energy efficient route to the data centre.
- In order to minimise the energy efficiency of cloud services (or resources), it is necessary to find and allocate suitable services, in terms of energy efficiency and power consumption, from lowest possible number of service providers. There has been a great deal of research into discovery and composition of cloud services, and tools have been developed with the power to serve users when they need it to use the services. A great deal of this research actually involved Infrastructure-as-a-Service (IaaS) layer and virtualisation types. However, searching for, allocating, and making provision plans for services that are energy efficient, is overlooked.
- While a number of heuristic IaaS solutions have been put forward, algorithms that will allocate resources in the most energy efficient way are still lacking. As time goes on, there is a development of hybrid cloud solutions combining IaaS and Platform-as-a-Service (PaaS) in a single cloud (OpenStack Heat is an example), and these are attractive because they make it possible for infrastructure and applications to be deployed together, but there is still not sufficient attention given to the energy efficient allocation of resources.

- There is no clear process for finding the most energy efficient services from multiple cloud providers that meet user needs.

1.1.5 Novelty of the thesis

This work makes a number of novel contributions, all of which have been, or are being, submitted to relevant research publications (Appendix I) and are summarised as follows:

- Defining and formalising the user-to-broker and data centre-to-broker models so that, the interconnection between the cloud user and a data centre, will be formalised by using a situation calculus model to define the logical state of the network. Once the interconnection is established and formalised, then the energy required for the transportation will be calculated.
- A high-end routing algorithm entitled Green Director (GreeDi), which acts as an intermediary bridge for directing the users' requests to the green data centres based primarily on using the most energy efficient route.
- An algorithm to rank cloud service providers by total energy consumption.
- An algorithm to enable the broker to search for the best possible match by seeking a combination of individual atomic service and lowest energy consumption.
- An algorithm to allow the broker to search predefined composition plans and their energy consumption in order to find the best possible match for user requirements.
- An algorithm to allow the broker to build a new optimal composition plan by selecting the most energy efficient services from the smallest possible number of providers.
- An algorithm to discover a predefined optimal composition plan from a single provider.
- An algorithm to create a dynamic optimal composition plan from multiple providers.

1.1.6 Research Scope

This research proposes novel support for achieving an energy efficient cloud, which is achieved by:

- A high-end routing model to achieve the full green cloud computing environment ambition while making sure the users' requirements, e.g. response time, are met. To create this model, we need to develop an algorithm that acts as an intermediary bridge for directing the user requests to the green data centres based primarily on using the most energy efficient route.
- A bin-packing based energy-efficient service model to find and integrate, from the largest possible number of service providers, the smallest possible number of services offering the highest level of energy efficiency. To enable this model, we need to create four algorithms to (i) Rank the Cloud Providers in a descending order based on total Energy Consumption, (ii) Find an atomic service that matches the user request, (iii) Find a predefined optimal composition plan from a single provider and (iv) Create an optimal services composition from multiple providers.
- A unique energy-aware multi-Cloud service composition model, which develops energy efficient composition plans through the integration of the fewest services globally from service providers. To achieve this model, we need to create two algorithms to (i) discover a predefined optimal composition plan from a single provider and (ii) to create a dynamic optimal composition plan from multiple providers.

1.1.7 Aim and Objectives

The aim of this work is to create and evaluate an energy efficient multi-cloud broker to address the gap mentioned in (Section 1.1.4). It acts as an intermediary bridge for directing users' requests to the data centres based primarily on using the most energy efficient route.

In addition, the broker manages the cloud services to find the most energy efficient service(s) that matches the user request; hence, achieving the most energy-efficient cloud computing environment. The primary objectives of this research include:

- Perform an extensive literature study on the cloud brokerage systems, the available cloud services delivery platforms to extract the components and features of these platforms and enable the managing of cloud services in a multi-cloud environment and achieve the energy efficiency.
- Create a new energy efficient services delivery algorithm that will be run in a multi-cloud environment to act as an intermediary bridge for directing the user requests to the data centres based primarily on using the most energy efficient route.
- Create an energy-efficient service algorithm to find and integrate the smallest possible number of services offering the highest level of energy efficiency based on a bin-packing approach.
- Develop an energy-aware multi-Cloud service composition model, which develops energy efficient composition plans through the integration of the fewest services globally from service providers.
- Test and evaluate the proposed algorithms against the well-known algorithms using network simulations.

1.1.8 Methodology

Objectives	**Methodology**	**Chapters**
Extensive literature study on the brokerage systems and routing solutions in the cloud environment	Conducting a literature review	Chapter Two and chapter three
Create a new services delivery algorithm for directing the user requests to the data centres based primarily on using the most energy efficient route	Using Situation Calculus model, Integer Linear Programming, Goal-oriented programing, and Dynamic programming approach	Chapter Four
Development of an Energy-Efficient Service Provision model to find and integrate the lowest possible number of services offering the highest level of energy efficiency.	Bin-packing approach Using Integer Linear Programming and IBM ILOG CPLEX Optimization Linear Solver	Chapter Five
Development of a service composition model, which develops energy efficient composition plans through the integration of the fewest services from globally distributed providers.	Using formal user requirements translation and transformation modelling and analysis	Chapter Six

1.1.9 Thesis Structure

The second chapter starts with the explanation of the cloud computing concept and its main players. In addition, it introduces the cloud service brokerage systems and their architectures resource management. Moreover, this chapter explains the cloud services models, cloud computing virtualization and the virtualization forms and categories. Furthermore, it describes the energy efficiency in cloud computing, the energy consumption in data centres and the energy efficient cloud resources allocation. The third chapter introduces the network

routing problem and the energy efficiency in cloud computing. It highlights the cloud broker selection problem and the energy consumpton issue in data centres and during the resource allocation.

Moreover, the fourth chapter intends to focus on the design and implementation of the energy efficient routing model for big data on the cloud model, which includes discussion about the situation calculus used to analyse the network topology. Moreover, linear, goal and dynamic programming modelling approaches will be discussed in this section. The fifth chapter presents the design and implementation of the bin-packing based multi-cloud model for energy efficient data-intensive applications and provides the main aim and objectives of this work. In addition, a detailed discussion on how the model works is presented in this chapter. The energy-aware service composition model and its design and implementation for multiple cloud applications, a detailed discussion of the proposed model and how it works is introduced in the sixth chapter. Finally, the research conclusion and the future work is presented in seventh chapter.

Cloud Computing

2.1. Introduction

The development of the proposed energy efficient multi-cloud broker in this thesis was inspired by previous works and is based on a number of paradigms, concepts and technologies which exist in cloud computing. Thus, as a convenient approach for presenting the overall state-of-the-art and related work, the literature review is structured into four main areas:

- Cloud computing environment: explaining the definition of the cloud computing concepts and its main players.
- Virtualisation: focusing on the virtualisation forms and the categories of virtualised servers.
- Cloud services: focusing on the classic cloud service models and the new hybrid one.
- Cloud service brokerage: studying the architecture of existing cloud brokers and the management of the resources when using a broker.

2.2. Cloud Computing

Cloud computing represents the fastest growing area in Information and Communication Technologies (ICT) at present. It gives users a cost-efficient way of obtaining different types of computing resources by paying only for what they use. The way cloud services are provided allows users and companies to outsource almost all ICT functions including hardware resources, services, and applications.

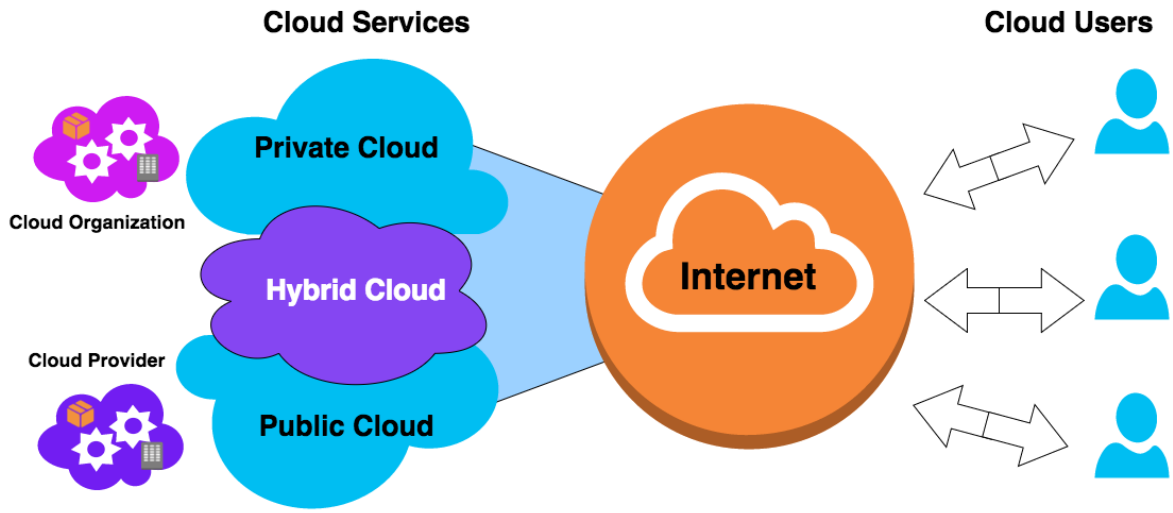

Figure 2.1 Cloud Computing concept (F. Liu et al. 2011)

The number and varieties of definitions of the cloud that can be found in the literature indicate, either explicitly or implicitly, the lack of a clear, complete and universally agreed definition of what this model comprises. Probably the most widely accepted one among the various definitions of cloud computing is the one put forward by NIST (National Institute of Standards and Technology) (F. Liu et al. 2011), which defines the cloud as:

"a pay-per-use model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services. It can be rapidly provisioned and released with minimal management effort or service provider interaction".

Accepting this definition permits extraction of a number of key features of cloud computing, such as:

- On demand automated resource provisioning and user self-provisioning (known as cloud self-service).
- Access to resources from anywhere and anytime on the network.
- Ability to pool resources, which can be assigned dynamically to suit each client's needs regardless of physical location.

- Immediate scalability (a.k.a. scale up/down/out) so that peaks in demands can be dealt with.
- Metering based resources used; similar to the other utility payment model enable the pay-per-use approach.

2.2.1. Deployment models of cloud computing

There are four hosting deployment models of cloud computing dubbed private, community, public, and hybrid. These models represent the categories of cloud computing environment and are distinguished from each other by the purpose, size, and access way, as follows:

Private cloud:

A private cloud is an infrastructure that is owned and managed by one single company, is used in that company's private network, and is not available to other users. It is also known as an internal cloud; the platform for cloud computing is implemented on a cloud-based secure environment that is safeguarded by a firewall, which is under the governance of the IT department that belongs to a particular corporate (Victories 2015). For example, a private cloud can be used by a financial company who will still need to benefit from some of the advantages of cloud computing and is required to store sensitive data internally.

Community cloud:

A community cloud is a type of cloud hosting in which the setup is mutually shared between many organisations that belong to a particular community, i.e. banks and trading firms. It is a multi-tenant setup that is shared among several organisations that belong to a specific group which has similar computing apprehensions. The community members generally share similar privacy, performance and security concerns. The main intention of these communities is to achieve their business related objectives. A community cloud may be internally managed, or a third party provider can manage it. Hence, it can be hosted

externally or internally. The cost is shared by the specific organisations within the community, therefore, community cloud has a cost saving capacity (Victories 2015). An example of this cloud is that a few organizations might require a particular application that resides on one set of cloud servers. Instead of providing every organization their server in the cloud for this app, the hosting company shares their environment to allow multiple customers to connect and segment their sessions.

Public cloud:

A public cloud is a large, high-volume and high performance infrastructure owned by one or more companies and provides IT services through the Internet to a multiplicity of consumers. This model is a true representation of cloud hosting. In this model the service provider renders services and infrastructure to various clients all around the world. The customers do not have any distinguishability and control over the location of the infrastructure (Victories 2015). A good real life example of this cloud is Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, Google AppEngine and Windows Azure Services Platform

Hybrid cloud:

Hybrid clouds combine public and private clouds, with a company's private cloud running those apps and/or managing that data that must be protected at all costs from external interference and from others among the company's apps running on a public cloud. It can be an arrangement of two or more cloud servers, i.e. private, public or community cloud that is bound together but remain individual entities. Benefits of the multiple deployment models are available in a hybrid cloud hosting. A hybrid cloud can cross isolation and overcome boundaries by the provider; hence, it cannot be simply categorized into public, private or community cloud. It permits the user to increase the capacity or the capability by aggregation, assimilation or customization with another cloud package or service (Victories

2015). An example of this type of cloud is an organisation that could deploy an on-premises private cloud to host critical or sensitive workloads, but use a third-party public cloud provider, such as Google Compute Engine, to host less-critical resources, such as test and development workloads. In addition, a hybrid cloud could also use Amazon Simple Storage Service to hold customer-facing archival and backup data. Moreover, it could use a software layer, such as Eucalyptus, to enable private cloud connections to public clouds, such as Amazon Web Services (AWS).

2.2.2. The main players in cloud computing

Like in any other utility provisioning model (i.e., gas, electricity and water), cloud computing has three main players, which are: providers, brokers and users. Each of these players plays a specific role and has a certain purpose from cloud computing. This section provides further details and definitions of these players.

Cloud Provider:

A cloud provider is meant to provide cloud infrastructure, through which cloud services are hosted. All responsibility for managing, controlling and maintaining cloud resources and for handling users' requests lies with the provider. A cloud computing provider's typical goal is to maximize its revenues with its employed pricing scheme. The cost will usually be based on a per-use utility model.

Cloud users:

A user is defined as an account with permission to log into the cloud application to use cloud services. A named user with this permission is counted towards the user limit and it could be individuals, companies, groups in which pricing will differ from one another and their main goal is to obtain the highest level of quality of service (QoS) feasible for a reasonable price.

Cloud Broker:

The Broker acts on behalf of both cloud users and providers, distributing user requests between providers according to which provider is best suited to meet that particular request. In this sense, brokers are acting as travel agents; the user tells the broker which application (type of holiday) is needed and the broker decides which service providers (holiday companies) offer the best deal. This analogy cannot be carried too far – travel agents are responsible for the resort or other type of location to which they send their customers, while brokers have no responsibility for where the actual computer would be located. Further details about the existing brokers and the way the broker works are set out in the third chapter.

2.3. Cloud Computing Virtualization

Server virtualisation has been used as a leading technology for sharing computing infrastructure for a while now. In fact, it was actually started back in the early 1960's by companies such as General Electric, Bell Labs and International Business Machines (IBM) (editor 2011). In the same vein, virtualization technology is what makes cloud computing possible. It works through the abstraction of physical resources by multiplexing a number of virtual resources onto one physical one as shown in Figure 2.2 Virtualization provides flexibility, convenient management of resources, elasticity in resource management, and isolation, while also permitting the coexistence on the same hardware of a number of heterogeneous services.

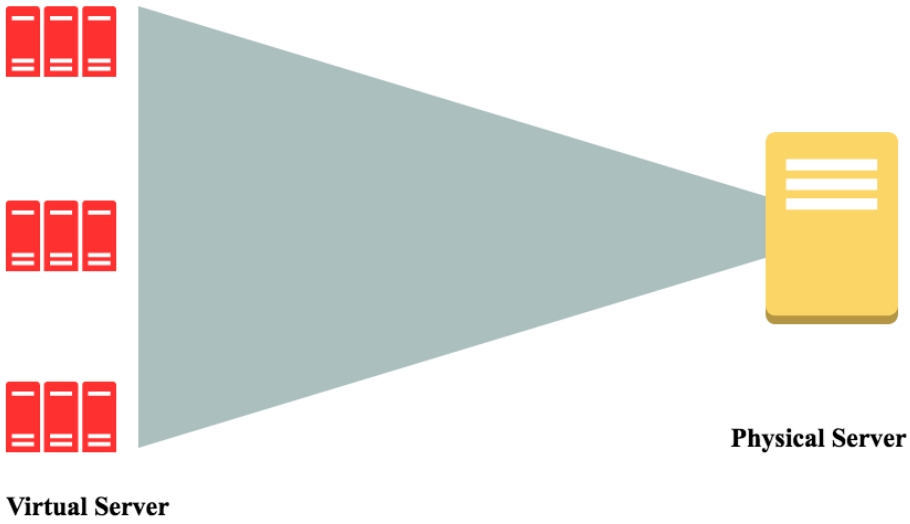

Figure 2.2 number of isolated virtual servers to run on one single actual server (editor 2011)

2.3.1. Virtualization Forms

Virtualization can be used in different shapes and forms in cloud computing infrastructure, including: server virtualization; storage virtualization; and network virtualization. All three rest on the same principle of abstracting physical resource through partitioning. Cloud computing's commonest method of resource abstraction is server virtualization, which – implemented in a variety of ways – permits a number of isolated virtual servers to run on one single actual server. Full virtualization, para-virtualization and OS-level virtualization are among the possible implementation methods. Full virtualization and para-virtualization both share the physical hardware by use of a hypervisor, but modify host and guest operating systems differently, as well as the way they interact among themselves, to make virtualisation happen. Virtualisation at the operating system level (OS-level), on the other hand, does not use a hypervisor; or virtual servers run the same host operating system, and this confers all functions that might otherwise be supplied by a hypervisor. It is therefore possible to divide server virtualisation into two types: virtualisation based on the hypervisor, and virtualisation based on the operating system or container. More information about this categorisation will be found in the next section.

2.3.2. Categories of server virtualisation

As discussed in the previous section, the way that is chosen to virtualise resources in cloud computing can be categorised in either of two ways: through hosted virtualisation with a hypervisor, or through container-based virtualisation.

2.3.2.1. Hypervisor based virtualization:

Hypervisor-based virtualisation was the way that virtualisation in the cloud was originally done. The hypervisor is a software layer tasked with managing physical server resources. Hypervisors in common use include: KVM (Kvm 2017), VMWare (VMware 2017), Microsoft Hyper-V (Hyper-v 2017), and Xen (Xen 2017). Virtual machines (VMs) can run a variety of operating systems including, but not limited to, Linux and Windows, whatever operating system may be run by the physical box that is hosting them. Introduction of an additional software layer by this type of virtualisation allows resource consolidation into virtualised servers (Srikantaiah, Kansal, and Zhao 2008) as well as providing live migration (Travostino et al. 2006) allowing VMs to be moved to other servers without being shut down.

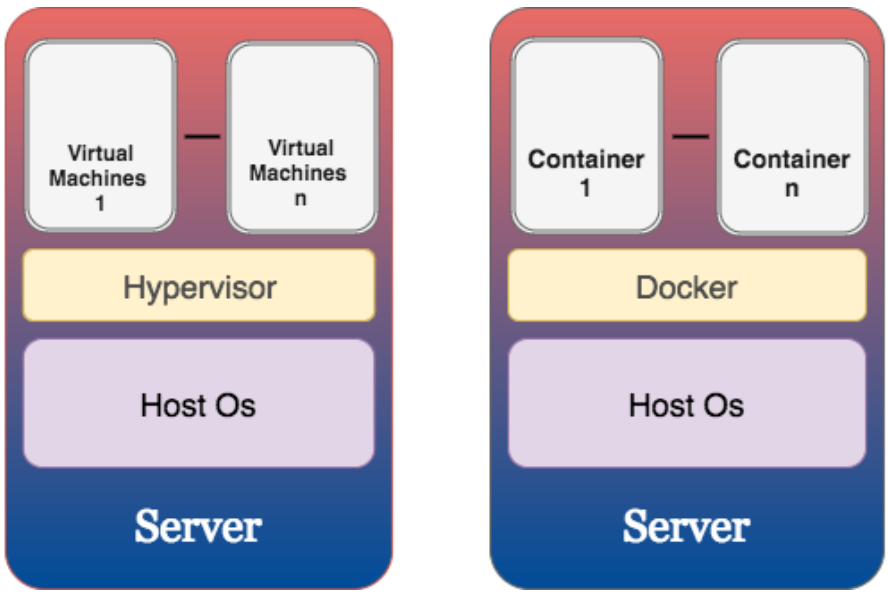

Figure 2.3. Container based virtualization vs hypervisor based virtualization (Travostino et al. 2006)

2.3.2.2. Container based virtualization:

Container based virtualization is an easy-to-use alternative to using hypervisors (Soltesz et al. 2007) (Xavier et al. 2013). This technology operates at the level of the operating system, permitting one server to host a number of isolated virtual environments; they differ from other VMs in that, based on a single shared operating system, they all run the same operating system as the host and cannot run different OSs. The differences between these two forms of virtualisation are shown in Figure 2.3 Container-based solutions include: Docker (Docker 2017), Linux containers (LXC) (LinuxContainers 2017), Solaris Containers (SolarisContainers 2017), Virtuozzo Containers (VirtuozzoContainers 2017) and OpenVZ (OpenVZ 2017).

In order to specify the best-fit virtualisation technology to use, it is important to study the desired system's features to have. For example, where flexibility and greater security are needed, or where there is a need to run different operating systems (Scheepers 2014), hypervisor based virtualization recommends itself. Otherwise, if the most important factor is performance, then container-based virtualisation comes into its own, being more manageable and providing performance at close to native level. The consolidation ratio is also higher and, because it allows one host to support a large number of instances, resource usage is more efficient. Container-based virtualisation brings with it portability, transport, and isolation at the process level across hosts.

The two categories of virtualisation do not exclude each other and they are increasingly being used together. Container-based virtualisation is frequently used to build PaaS environments, while IaaS cloud services really need hypervisors, so that having resort to both solutions means that complex services can be deployed over hybrid IaaS/PaaS cloud providers to bring together applications and the underlying infrastructure in such hybrid

solutions as Proxmox (Proxmox 2017), which enable both technologies to be supported on one physical server.

2.4. Cloud Services

Cloud service models define the way in which services are provided to users. There are two model types: classic cloud service models and new hybrid ones.

2.4.1. Cloud service models

There are three types of classic cloud service models as shown in Figure 2.4: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

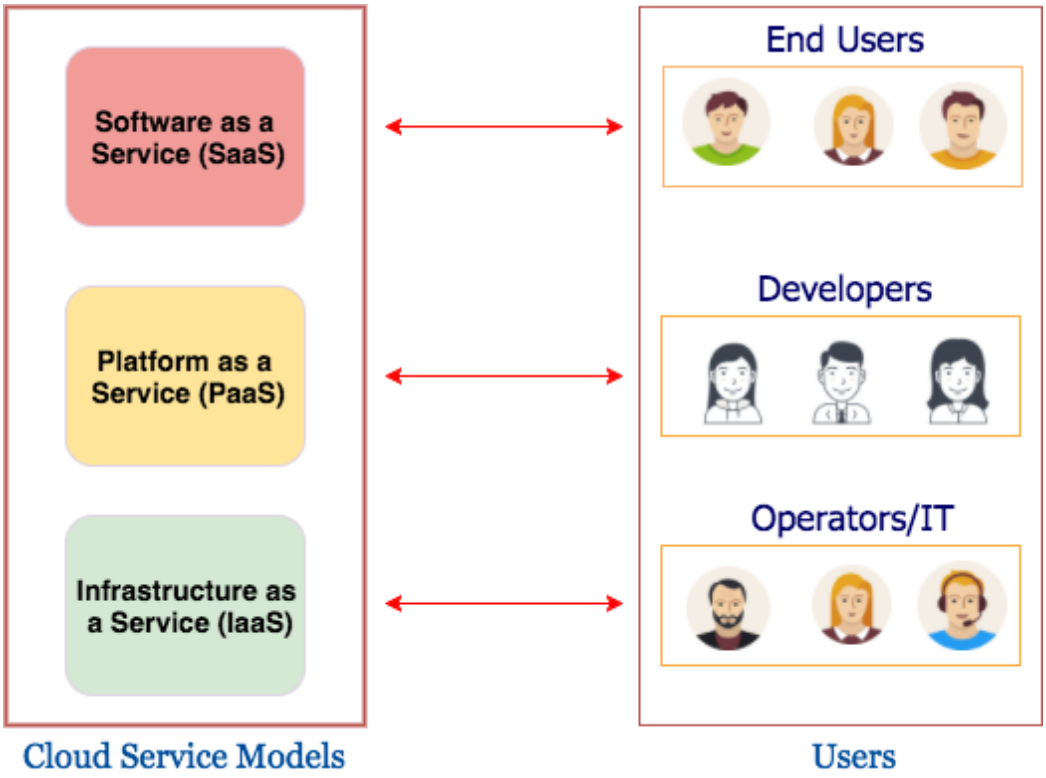

Figure 2.4 three types of classic cloud service models (Kubernetes 2017)

2.4.1.1. Infrastructure as a Service (IaaS):

IaaS is the cloud services model that is easiest to understand, because in effect it sums up exactly what the cloud is about: provisioning and delivering such resources as virtual machines, physical servers, network and storage. Where companies previously invested in

their own infrastructure, they usually have to buy more than, most of the time, what they need because of peaks in demand. However, with the cloud computing IaaS model, they can now rent resources on-demand, based on their use and need using the pay-per-use model. An IaaS user can access the lowest level in the stack directly and build an application environment from the ground up. One such IaaS Cloud is Amazon Elastic Compute Cloud (EC2) (Amazon 2017).

2.4.1.2. Platform as a Service (PaaS):

PaaS is rather a more complex, higher level service than IaaS. In this level, software platform and an application development environment are provided, where users can build cloud applications and subsequently deploy and manage them without having to concern themselves with the details of the technology and infrastructure they are hosted on. Instead of waiting for time on an in-house machine to become available, developers can get on with the job straightaway, paying only for resources they use, as mentioned above. Well-known PaaS platforms include Google App Engine (GoogAppEngine 2017) and Microsoft Azure Services (MicrosoftAzure 2017).

2.4.1.3. Software as a Service (SaaS):

The highest-level Cloud service model is SaaS. The work is done for the user and provided through the Internet. Not only do the providers manage the infrastructure; they also provide and run application software, which can be accessed by users in the same way as those would access a locally hosted application. It is not necessary for users to know anything about the cloud and its associated technology – they do not even need to know that they are running a cloud-based application, but rather they need to focus on how to use these application to achieve their tasks. Facebook and Salesforce.com, for example, are SaaS applications. Large numbers of other commercially available systems including Google

Documents (GoogleDocs 2017) and Google Apps (G suite) (GSuite 2017), are some other examples.

2.4.2. New hybrid service models

As the users request more flexibility and more control in order to deploy applications in the way they choose and with having the control on both infrastructure and programs, the classic three layer concept has often been a subject of discussion. The consolidation of IaaS and PaaS is expected, while there is a steady blurring of divisions between cloud services so that IaaS and PaaS increasingly work together and are presented as two sides of the same coin, while new hybrid Cloud providers are emerging to provide a way for users to bring together a variety of services.

It is in the interests of such companies as Amazon, Microsoft and Google, as well as being clearly in the interests of users, but those users when requesting Cloud services, should not need to think about whether it is IaaS or PaaS that they need. One such emerging cloud service is Kubernetes (Kubernetes 2017), an offering from Google that combines IaaS and PaaS. The IaaS provider, Openstack (OpenStack 2017), is adding PaaS features in order to provide seamlessly combined IaaS-PaaS services by orchestrating Docker (Docker 2017) containers through the use of Openstack Heat.

451 Research (Jay Lyman 2014), is a global analysis company with a particular interest in IT innovation at the enterprise level. It says ”Although it is maturing in technology and market, PaaS is getting squeezed between consolidation with IaaS and heavy use of SaaS. PaaS will most likely survive as a category, but not necessarily as we know it today”, which indicates the new category to combine IaaS and PaaS in order to enable users to create a single continuum of services.

2.5. Cloud Service Brokerage

NIST and Gartner have provided a definition of Cloud Service Brokerage (F. Liu et al. 2011; Gartner 2013), which revolves around a three-pronged categorization. According to the two organisations, cloud broker refers to a middle entity, which oversees the negotiations and relationships between cloud consumers and cloud providers; and manages the use, performance and delivery of cloud services (Forrester 2012). The focus of Gartner customisation and NIST intermediation is on promoting the existing service. Gartner integration and NIST arbitration share a similarity that is a reflected integration of diverse systems and flexible mediation. As shown in Figure 2.5, there are three cloud service brokerage roles:

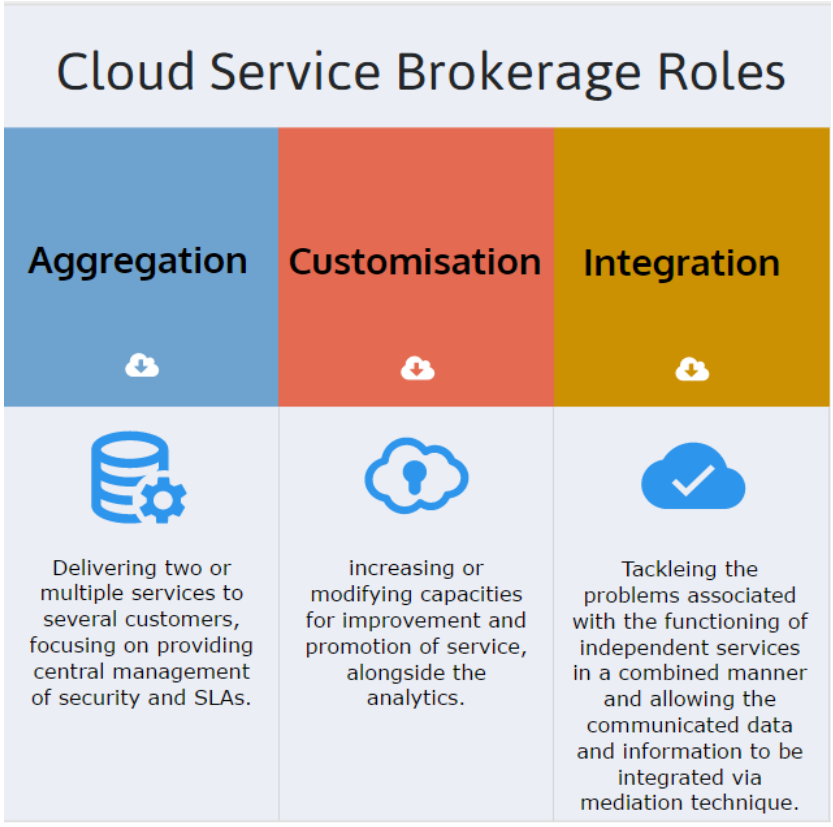

Figure 2.5 cloud service brokerage roles to manage the broker

Aggregation focuses on the delivering of two or more services to several customers. However, this does not involve provision of new customisation, integration, and functionality; but rather it focuses on providing central management of security and SLAs.

Service aggregation models play an important role in the broker systems as they help to deploy customer applications across multiple cloud providers (L. Zhang, Fowley, and Pahl 2014). Some jobs across multiple cloud providers that offer similar or different types of services can be aggregated to meet user requirements. Sometimes, consumers may have specific time and limited budget and need their application to be distributed across multiple cloud providers to meet their requirements, in which case the aggregation helps the broker to achieve this task.

Customisation involves increasing or modifying capacities for improvement and promotion of service(s), alongside its analytics. Multiple services can be bundled and customised into one or more services that are contracted directly the cloud users. It enables the organisations to add more functionality using their own processes of choice rather than being limited by a vendor-specified approach. Moreover, customisation allows users to control the service based on their needs.

Integration tackles the problems associated with the functioning of independent services in a combined manner. It usually involves combining processes, resources and/or services vertically across the same cloud provider, or horizontally across multiple cloud providers. Conventional methods such as orchestration, mediation and transformation offer solutions to compose these resources together. Generally, the integration process allows the communicated data and information to be integrated via a mediation technique where the cloud broker can build services on top of the services, such as management capabilities or additional security. In addition, service discretion plays an essential role in cloud brokerage systems especially in the integration process. It enables the system to discover, deploy, and manage multiple services (Sun, Dong, and Ashraf 2012).

2.5.1. Cloud Service Broker Architectures

Cloud brokerage solutions are based on existing SaaS, PaaS, IaaS, cloud platform, and virtualisation. According to Fowley et al. (Fowley, Pahl, and Zhang 2013), there are three architectural patterns identified:

Cloud Management: supports monitoring, providing, deployment, and designing cloud resources. For instance, using management portals; this constitutes an expansion of the central lifecycle management (LCM), incorporating tracking features and interactive graphical forms. Rudimentary characteristics for integrating compatible services may be offered. A management layer is usually located within cloud architecture to management, which enhances scalable and effective provisioning within the following platforms.

Cloud Broker Platform: is used for supporting the types of broker activity such as integration, customisation, and aggregation that requires a specified language for description of services such as unified service description language (USDL) in a balanced manner and for definition of the integration mechanism. This originates from the common broker pattern in software design, applied on a cloud environment. To obtain the right solutions and to enable the customisation of applications in order to meet the end-user needs, each cloud service broker has a partnership with several computing platform providers.

Cloud Marketplace: builds up on broker platforms and it brings customers and providers together. Additionally, description of service for integrated and core services are critical in enhancing technical and functionality quality features. The second aspect that requires facilitation is trust. Notably, apps marketplaces are ubiquitous, thus the marketplace pattern reflects future cloud-specific marketplaces.

2.5.2. Resource Management in Cloud Broker

The support of diverse service consumers by the cloud brokers is based on its ability to integrate resources from many cloud providers. The users access the Web services hosted by

the cloud providers over the Web interface. Basically, the cloud provider aims at satisfying the SLA set with the client for the hosted services through the utilization of the minimal amount of hardware resources that is received from the cloud broker (Nair and Porwal 2010). On the other hand, the cloud broker targets profits improvement through the leasing of the optimal percentage for the requested computing resources provided to the consumers. Additionally, the computing environment has unpredictable differences such as software and fees leading to an excess the arrival rate of web request more than the expected value. Thus, it becomes necessary for the cloud broker to be asked for additional resources by the consumers. Therefore, it is important for the consumers to negotiate with the cloud provider for the resources by relying on pricing schemes that support in maintaining the SLA as well as minimizing the cloud resources costs (Nair and Porwal 2010). Furthermore, additional resources may be required by the multiple consumers via the cloud broker. Thus, it may lead to competition over the limited resources by the cloud providers.

The researchers now focus their attention towards the solutions addressing resource management challenges in data centres (Beloglazov and Buyya 2010). Mostly, the approaches fail to address the problems of management of resources within the framework of the organisation. The reliance on the cloud brokers helps in the integration of many resources from numerous cloud providers, which requires the exposure of the requirements to orchestrate and compose these resources. The recent literature has indicated that the efficient operation and management of the cloud computing environment is essential because the framework for federated cloud computing has increased in demand. Therefore, the cloud brokers provide the best hope for dealing with the multi-cloud environment complexities. In that regard, it emphasizes the need for efficient management and allocation of the resources in the model of relying on cloud brokers (Rogers and Cliff 2012).

2.6. Summary

This chapter presented a literature review on a number of paradigms, concepts and technologies existing in cloud computing. It introduced cloud computing and explained the definition of the cloud computing concepts and its main players. Moreover, the cloud computing virtualisation and its forms and the categories of the servers were explained. In addition, this chapter highlighted the cloud services and their three classic cloud service models and the new hybrid one. Finally, the architecture of the cloud broker and the management of the resource when using the broker was explained.

Network Routing and Energy Efficiency in Cloud Computing

3.1. Introduction

The main objective of the service brokers is to direct the user requests to the best datacentre that has optimum performance. For example, the service broker policy must choose a data centre taking into consideration multiple factors such as availability, cost, and time. Online services have encouraged service providers and data centres to offer hosting in every geographical region, which obviously resulted in a high increase in network traffic and a matching rise in the energy consumed by the vast infrastructure (e.g., servers, routers, and switches). This chapter discusses the following issues:

- Network routing: presenting the routing problem in cloud computing when a broker selects a service. In addition, other techniques such as segment routing and sensor networks routing algorithms will also be discussed.
- Energy efficiency in cloud computing: explaining the energy issues in the cloud and its resources.

The chapter concludes with an outline of the main requirements for an energy efficient brokerage solution in multi-cloud environments, which paves the way to the design, development and evaluation of the proposed broker in following chapters.

3.1.1. Cloud broker problem

As mentioned above, the service broker relies on defined routing policies to choose the best possible data centre to serve/execute the coming job(s). The three standard routing policies are: (1) network latency-based (i.e., service proximity based routing) (Limbani 2012)(Sharma 2014), (2) response time-based (i.e., performance optimized) routing (Q. Zhang, Cheng, and Boutaba 2010) and (3) Dynamic load-based (i.e., dynamically reconfigure) routing (Rekha P.M. and Dakshayini 2014). In service proximity based routing, the broker selects the shortest path from the User Base (UB) to the data centre based on network latency, so transmission latency is considered by the service broker when routing

the traffic to the closest datacentre. The result of this routing policy is the overloading of the nearest data centre and communication channel to it, as it does not consider the channel bandwidth. Whereas in performance optimised routing policy, the broker selects the best path based on performance of all datacentres, thus the traffic is directed to the datacentre based on the best response time. This will be generalised as the status of any other datacentre. As such, any data centre with a zero-current load, will not be chosen unless a specific amount of time is known (i.e., Cool-Off-Time). This might render the data centre "*idle*" with no jobs assigned to it even if it was the closest to the UB (i.e., least latency) and highest bandwidth network path available (Rani, Chauhan, and Chauhan 2015). Alternatively, the dynamically reconfigured routing has similarities with proximity based routing, however the broker scales application deployment is based on the load faced (Limbani 2012) (Wickremasinghe, Calheiros, and Buyya 2010)(Semwal and Singh Rawat 2014).

As can be seen above, there are some drawbacks that are shared among the three service broker policies. For example, in service proximity based routing, the service broker does not include network bandwidth or the request data size, which might cause a significant degradation in overall performance, particularly in the case of big data or where there are multiple requests which share bandwidth and the same communication channel (i.e. file or gaming servers) (Rekha P.M. and Dakshayini 2014). Alternatively, performance optimised routing has similar problems as the service proximity routing policy. It takes into consideration server load by referencing previously performed jobs and choosing them accordingly, irrespective of job size and network bandwidth (Sharma 2014) (Sarfaraz Ahmed 2012). Lastly, dynamically reconfigured routing is inefficient if region numbers and quantity of data centres are limited; as it scales application deployment based on current load (Rekha P.M. and Dakshayini 2014).

Due to these issues, there has been additional research work aim at improving brokerage polices in terms of bandwidth, cost, workload, response time, and processing time. Limbani in (Limbani 2012) suggest a service broker policy which aims to choose the data centre with the lowest cost within the same UB region. Such a policy is not efficient in choosing the lowest cost data centre though. Yet, it still lacks competence as it does not consider work load, bandwidth, file size, and response time (Limbani 2012) (Rekha P.M. and Dakshayini 2014). Thus, (Semwal and Singh Rawat 2014) suggested a new policy that selects the data centre with the highest configuration to optimise the response time. Although the goal was reached, however, it also increased overall cost if the data centres were processing huge quantities of data (Deepak Kapgate et al 2014) (Mishra, Kumar, and Sreenu Naik 2014) (Rekha P.M. and Dakshayini 2014) (Semwal and Singh Rawat 2014).

Deepak *et al.* in (Deepak Kapgate et al. 2014) suggested a data centre selection algorithm based on how many times the data centre is selected in accordance with its memory requirement and processing capacity of the upcoming requests. The author focuses mainly on the reduction of the associated overheads and service response time whilst also improving overall performance. Given that the proposed algorithm can improve performance of the existing proximity algorithm, it still suffers an increase in the overall cost (Deepak Kapgate et al 2014). To get around this issue, Sharma, (2014) applies the Round-Robin load balancing policy so that the workload is distributed among multiple available data centres in the same region. The Round-Robin load balancing policies try to equalise the overall cost of the data centres. Vaishali Sharma results have shown resource utilisation efficiency under the proposed environmental simulation. But this might not necessarily be the case in all instances if the data centres have different configurations (Sharma 2014). As a consequence, Mishra, Kumar, and Sreenu Naik, (2014) suggested an extended Round-Robin service broker algorithm, which tries to distribute the coming

requests based on data centre ratings to enhance overall cost with a minimal response time. The suggested method is an improvement upon the random selection algorithms. Yet, if there exist some data centres that are faster than others, they will be frequently selected, and therefore get overloaded (Mishra, Kumar, and Sreenu Naik 2014) (Sharma 2014).

3.1.1.1. Load Balancing Algorithms

Appropriate load balancing techniques do not only assist in cost reduction, but also make enterprises meet user satisfaction (Nidhi Jain 2012) (Ali Alakeel 2012). Scalability is therefore a crucial feature of cloud computing, as cloud computing also gets affected by load balancing. Therefore, efficient load balancing, across multiple cloud data centres, helps in reducing energy consumption through improvement to resources utilization and hence overall distributed system performance.

In a cloud computing environment, load balancing algorithms are usually divided in two categories: Static Load Balancing Algorithms and Dynamic Load Balancing Algorithms (Randles, Lamb, and Taleb-Bendiab 2010).

Static Load Balancing Algorithm

Static load balancing algorithms assign tasks to the nodes, based only on the node's ability to process new requests. However, they do not take into consideration any dynamic changes (e.g., server load, server availability and distance between user and server) of these attributes at run-time. Additionally, the algorithms cannot adapt to load changes during run-time. This process is solely based on existing knowledge of a node's storage capacity, memory, processing power, and the most recent known communication performance. Round Robin (RR) and Weighted Round Robin (WRR) are most the commonly used static load balancing algorithms used in cloud computing (Randles, Lamb, and Taleb-Bendiab 2010). Round Robin algorithm does not consider the distance between clients and servers, server load,

server availability. In this algorithm, server selection for processing requests is done sequentially. The main problem with such an approach is inconsistencies in server performance, which can be overcome by WRR. In WRR, the weights are added to servers depending on the amount of traffic that is directed to servers. However, for long time connections it leads to load tilt (Randles, Lamb, and Taleb-Bendiab 2010).

Dynamic Load Balancing Algorithm

Dynamic load balancing algorithms consider both knowledge based on existing information gathered about network nodes in the cloud, and run-time properties which are gathered as the selected nodes process task components. The algorithms are given the tasks and might reassign them dynamically to the nodes based on the attributes gathered and calculated. Yet, they have accuracy and might result in greater efficiency of load balancing than the above static load balancing algorithm.

Least Connection (LC) and Weighted Least Connection (WLC) (Randles, Lamb, and Taleb-Bendiab 2010) are both commonly used dynamic load balancing algorithms. In the case of LC, the total number of connections on a server is identified at run time and the incoming requests are sent to the server with a smaller number of connections. But, LC does not take service capability into consideration, or the distance between clients and servers. Whereas WLC takes into consideration both the weight assigned to the service node $W(S_i)$ and the current number of connections of the service node $C(S_i)$ (Tian Shaoliang, Zuo Ming 2007) (Qi 2006). The issue with WLC is that, as time progresses the static weight cannot be subjected to correction, and the node will deviate from actual load conditions, which will result in imbalances in the load.

Ren, Lin, and Zou, (2011) made a prediction-based algorithm called exponential smoothing forecast. This is based on Weighted Least-Connection (ESBWLC), which can deal with long-connectivity applications. In this algorithm, the server load is calculated from various parameters such as size of disk occupation, number of connections, memory usage, and CPU utilisation. The load per processor (Load/p) is then calculated and the algorithm uses (Load/p) as an historical training set, then establishes a prediction model and makes a prediction of the value of the next moment. However, the ESBWLC algorithm does not consider distance between servers and clients, network delay and other factors. Kapgate and Narnaware, (2013) suggested Extended-ESBWLC, which can overcome this limitation. The Extended-ESBWLC algorithm calculates directly the client-side response time. This response time is stored for future reference. The response time at time instance 't+1' can be predicted by using the current response time at time instance 't' and the previously predicted response time for time instance 't'. This static service broker algorithm provides an improvement in results in terms of reduction in data centre loading, reduction of data centre request timing, and cost reductions of VM and data transfer. The author improves upon the service broker algorithm, called the service proximity service broker.

3.1.2. Segment Routing

Segment Routing (SR) is a network technology that offers a new method of packet forwarding that minimizes the need for keeping large numbers of network information states and therefore helps to overcome the TCAM deficiency problem (M.-C. Lee and Sheu 2016). A network makes use of an interior gateway protocol (IGP) for every pair of host communications within the same Software Defined Network (SDN) and Segment Routing (SDN/SR) domain. For example, the open shortest path first (OSPF) protocol allows, by default, routing to the destination. SR (Clarence Filsfils, Stefano Previdi 2017) uses a "node segment" which represents the activities a packet must follow to take whichever route is

shortest. To put it another way, each node in the same SR domain retains a "node segment" information link to all of the other nodes in its forwarding table, this is because the default rules and the node segment signifies global awareness. Additionally, SR uses an "adjacency segment" so that it can control traffic. Adjacency segment represent the action, where the packet must transfer to a particular data link with an adjacent node. Hence, adjacency segment represents local awareness. Yet, different to the node segment, each of the nodes needs only to install its local adjacency segment rules in the forwarding table. The combined adjacency segments and node segments form a sequential list of labels which, using the SDN controller, are applied to the packet header; they are instantly reflected as the preferred traffic path. Thus, SR allows several orders of scaling gains, because it does not hold any state for the flows in the transitional devices. Yet, SR also presents another problem, as it uses multi-protocol label switching (MPLS) label field when placing the segment labels. Therefore, SR might need a larger packet header, which causes a reduction in the available bandwidth.

The subsequent paragraphs show a summary of previous research related to the SR technology. Firstly, the authors in Bhatia et al., (2015) contemplate the issue of how to determine the optimal parameters for SR in online and offline cases. The authors suggest a traffic matrix oblivious algorithm for the offline case, and an alternative algorithm for the online case. In the online case, the network features a centralised controller, which uses an online method to answer the SR problem, as performed in the SDN-based environment. The authors provide formulae and linear programs which define the SR optimal parameters. The paper gives focus to the determination of the optimal parameters as traffic split values. Such values are applied to reduce as far as possible the worst-case link utilisation by taking into account equal-cost multi-path routing (ECMP) in offline cases. The traffic split values can also be used to minimise the number of request rejections in online cases. Yet, this research

is devoted to the design of an efficient routing algorithm to ensure enhanced network performance (e.g., better network throughput and rejection rate) and to pay attention to seeking the shortest path without equal-cost multi-path routing (ECMP). This algorithm contemplates the link residual bandwidth and link criticality when evaluating the link weight. Additionally, it places a limit on the maximum route path length to lower bandwidth consumption by the network.

The issue of energy consumption during deployment of large-scale distributed infrastructures was raised by the authors in (Carpa, Gluck, and Lefevre 2014). They have proposed that using SR-based energy-efficient traffic engineering can lower energy consumption of backbone networks. The network is able to selectively switch off a subset of links via the SDN approach. This technique was implemented by the authors in the OMNET++ simulator which decides dynamically the quantity of power-on links, which represents an energy saving. The objective of the authors is to find out network device status, irrespective of if there is a requirement to transfer data. However, the author intends to construct a bandwidth-satisfying path, which leads to an increase in total network throughput, and can lower the rate of request rejection, instead of energy consumption decrease.

An architecture, which can integrate the SDN paradigm with SR-based traffic engineering, is introduced by the authors in (Davoli et al. 2015). They paid attention to the issues of mapping computed paths onto SR paths. They have suggested an SR path assignment algorithm which intends to discover the shortest list of segments which corresponds with the desired path.

The authors in Lazzeri et al., (2015) propose a segment list-encoding algorithm to express a given path in order to minimise the segment list depth in SR-based networks. This algorithm provides a method for ECMP-aware shortest path computation, which is subjected to a

considered set of multiple constraints. This research also takes into account the label stack depth (LSD), and suggests a routing algorithm, which will lower the additional packer header cost, as this is caused by label stack depth. Additionally, it take into consideration the balance of traffic load in our routing algorithm, with the intention of improving performance related to the rejection rate and network throughput. Subsequently, it focuses on the task of improving the routing algorithm instead of solving the segment list computation problem.

The SR technology in (Sgambelluri et al. 2015) is applied to a multi-layer network test bed. The authors have designed an SDN-based SR solution, which controls network edge nodes for configuring the label stacks. The paper also shows scalability tests for varying label stacking conditions. The central issue featured in the work is the demonstration and implementation of SDN-based SR. The authors do not engage in discussion regarding the routing algorithm for the SDN controller to configure the route in edge nodes.

3.1.2.1 Traffic engineering

A lot of the existing work deals with traffic engineering as a vital tool for network performance upgrades, which works by direct traffic particularly through a finite competitive network resource. Traffic engineering works to configure the routing scheme so that traffic is controlled and routed across the network in order to efficiently use network resources and optimise network performance. It is crucial that traffic engineering's methods can use the measured "*traffic matrix*" for the management and diagnosis of congestion on the network. The traffic matrix represents the traffic volume between sets of source-and-destination pairs over a particular amount of time, and this element is a key factor in network planning. Traffic matrix estimation means simultaneously gathering information on routing and measurement of traffic flow. SDN permits network measurements, which are more dynamic and can also determine the precise and timely traffic matrix due to OpenFlow and the centralised controller (Malboubi et al. 2014)(Tootoonchian, Ghobadi, and Ganjali

2010). Therefore, when studying traffic engineering, it is possible to obtain predictions of future traffic trends via the estimated traffic matrix, where we can also discover good routing configurations (Han and Moutarde 2012).

In (G. Apostolopoulos,D. Williams,S. Kamat, Lucent et al. 1999)(Zheng Wang and Crowcroft 1996), the routing strategies employed include the shortest-widest path (SWP) and the widest-shortest path (WSP) algorithms. Path width represents total available bandwidth, and the length usually aligns with the number of hops. Thus, the aim of the WSP algorithm is to choose the shortest path, which consists of the largest quantities of residual bandwidth. The dominant result in the application of the WSP algorithm is network cost reduction, as the algorithm focuses on resource preservation by selecting the minimum hop count path. The SWP algorithm selects the maximum amount of available bandwidth from the source node to the destination node. When there are multiple paths, which have similar maximum available bandwidth, the algorithm chooses the shortest path. Therefore, using the SWP algorithm results in load balancing, this is because path selection has the maximum available bandwidth along all possible paths for each of the requests. The use of the SWP algorithm brings a risk of increased network cost, because the widest path usually leads to a longer hop count path, this uses more resources and therefore lowers network throughput.

(Kodialam and Lakshman 2000), the concept of interference was introduced by the authors, this takes into consideration that routing a flow along a specific path can lower the maximum flow between some of the other pairs. This concept advocates that a new routed connection can follow a path, which does not permit more interference to any of the other paths where the links might be vital to traffic demand for the other host pairs in the future, for the other host pairs. The authors have proven that the interference problem is NP-hard, and have suggested a heuristic minimum interference routing algorithm (MIRA) which maximises the minimum-maximum flow between all the other source-destinations. The

authors have demonstrated that the MIRA's rejection number is lower than the other algorithms which do not take into consideration minimum interference criteria. The MIRA also have some limitations. E.g., the intricacies of repeated maximum flow computation requires $O(VE^2)$, as well as the MIRA focuses on interference effects on only the critical links, therefore ignoring the non-critical links (where V is the number of switches and E is the number of links between switches). Thus, the length of the routing paths can be long enough to render a path all but unusable.

3.1.3. Routing algorithm to balance energy consumption

A lot of research work has been conducted on multi-path routing protocols, but one of the main challenges for researchers is balancing energy within sensor networks. The technique of network flooding, which can find routes between nodes, is used in many of the proposed routing schemes for WSNs. The authors in (Perkins, Charles E.,Elizabeth M. Belding-Royer 2003) employed a flooding technique to discover the shortest route, however it is not always useful for energy consumption when the routes detected do not have enough energy to send data between source and destination. Else, it can be assumed that node energy capacity is a vital resource and needs to be consumed in such a manner as to ensure long network lifetime.

In AlShawi et al., (2012), the authors have suggested a new routing method for Wireless Sensor Networks (WSNs) to increase the lifetime of the network by using both an *A-star* algorithm and *fuzzy approach*. The suggested method intends to take an optimal routing path from source of destination by favouring the lowest traffic load, minimum number of hops, and highest remaining battery power, and balancing them to extend the lifetime of the network as much as possible. But, such a method imposes a significant overhead in terms of computational complexity and communication. The methods in Montoya and Donoso (2013) have implemented a multi-path technique which balances power consumption to extend

WSN lifetime. The suggested heuristic technique uses a mechanism which constructs a path, and chooses the next communicating sensor node according to Received Signal Strength Indicator (RSSI) level and distance. In addition, whilst a path is constructed, the algorithm can avoid cycles and therefore has the capability for path reconstruction when the discovering route has arrived at a leaf. When several paths from the same target arrive at a particular sink, then the node employs an energy balancing strategy which decides on the flow quantity to transmit to each path.

Two approaches are presented by the authors in Kacimi, Dhaou, and Beylot, (2013). Firstly, a method of traffic load balancing is employed to optimise node energy consumption in a grid topology with a base station in one of the corners. Therefore, a distributed heuristic algorithm is suggested in order to combine transmission power control with load balancing to find ideal traffic proportions between the nodes, which ensures energy consumption is balanced. This method only works for a grid network.

The authors in Ming Lu and W. S. Wong, (2007) suggest that an Energy-Efficient Multi-path Routing Protocol (EEMRP) can search multiple node-disjoint paths and uses a load balancing technique to allocate traffic over each of the paths. Both node residual energy level and the number of hops are taken into consideration and can be incorporated into the link cost function. This link cost function is employed by the node to choose the following hop in the path search phase. Further, because EEMRP is only responsible for data transfer delays, successful path reliability is frequently limited.

In T. Liu, Li, and Liang, (2012), the authors suggest that an unequal Clustering Algorithm (EBCAG) and Energy Balancing can partition the nodes into clusters of unequal size, each of the sensor nodes keeps a gradient value, this is defined as the minimum hop count to the sink. Cluster size is determined by the cluster head gradient value, and the data obtained from all of the cluster members needs to follow the descending gradient decision so that it

can reach the sink. Yet, such an approach is based on a WSN with uniform distribution. In real-world applications, the uniform sensor distribution approach might not be practical or practically possible. In a similar manner, the authors in Ducrocq et al., (2013) have used a technique of balancing energy consumption in clustered WSNs, a clustering algorithm which then selects a sensor as cluster head in regard to density, node degree, and remaining energy. The algorithm offers the chance for each of the sensors to become a cluster head which balances energy consumption in all of the sensors. The work presented in T. Liu, Li, and Liang, (2012) attempts to maximise the time for all nodes to remain alive so that application requirements can be satisfied. In Jemili, Tekaya, and Belghith, (2014), a proposal is made for a Fast Multi-path Routing Protocol for wireless sensor networks (FMRP). The idea is that selecting multiple paths avoids the effect of inter-path interference, without the need for a costlier step. The protocol proposed allows establishment of node disjoint paths between source nodes and the sink while at the same time reducing interference and collision impact. FMRP offers a mechanism for efficient route discovery, this permits the avoidance of selecting highly correlated paths.

In Banimelhem and Khasawneh, (2012), the authors have suggested a Grid-based Multi-path with Congestion Avoidance Routing protocol (GMCAR), where they use the idea of dividing the sensor network field into grids. In each of the grids one of the sensor nodes is chosen as the master node, this has the responsibility of delivering generated data by any of the nodes in that particular grid, and the data routing from master nodes in neighbouring grids. For all of the master nodes, there are many diagonal paths which connect the master node to the sunk and the hop count is stored as a routing entry in the routing table of each node which uses them for routing decisions. The procedure is only effective for grid sensor networks.

3.2. Energy efficiency in cloud computing

The increasing demands on cloud computing resources has led to a substantial growth in the data transferred between cloud computing parties (i.e., providers, data centres, users, and services), network traffic and the energy consumed by the huge infrastructure of cloud computing, which is needed to meet the users' requests quickly and effectively. The cloud network traffic is forecasted to increase about threefold between 2013 and 2018 (Larumbe and Sansò 2012). Routing big data between the cloud computing parties requires a high bandwidth connection, which consumes larger amounts of energy (Chang Ge, Zhili Sun, and Ning Wang 2013) than just processing and storing big data on cloud data centres, and thus, producing high carbon dioxide emissions.

3.2.1. Energy Consumption In Cloud Datacentres

If energy efficiency is to be improved in the cloud, the way in which power is distributed in typical data centres and the actual power flow must both be understood. More than 50% of the electrical power consumed goes to feed the IT loads.

A report submitted to Congress by the Environmental Protection Agency EPA on Server and Data Center Energy (Brown et al. 2007) said that 80% of total IT and 40% of the data centre's total power consumption was consumed by servers, with the rest of the power going to transformers, air conditioners, cabling, pumps, lighting and other such devices. While power consumption by cooling equipment is significant, it is proportional to IT power consumption. Free cooling is among technologies that big companies such as Facebook and Google use due to their ability to reduce the amount of power that cooling consumes. For mechanical refrigeration methods, they substitute the use of naturally cool air or water, and this has resulted in a very large decrease in the amount of electrical power that cooling demands. In certain climates (Canada and north of Europe), there can be no refrigeration at all, so that the money saved reaches 100%.

3.2.1.1 The main factors leading to waste-to-energy

As we have seen, the heaviest consumers of power in cloud data centres are servers. There are a number of reasons for this:

Reduced utilisation of servers:

As data centres grow larger, there is a steady and continual rise in the number of servers, and most data centre servers are used less than they could be. America's Natural Resources Defense Council (NRDC) say that between the years 2006 to 2012 there was no growth in the average server utilisation (which averages somewhere between 12% and 18%) (Whitney and Delforge 2014). Nevertheless, the report also says that between 60% and 90% of peak power is taken by the servers. If virtual servers are consolidated on a reduced number of hosts, then the same applications can be run while consuming much less power. Increasing the utilisation of servers can greatly reduce both the number of servers needed and their overall power consumption.

Idle power waste:

Data centre servers spend 85% to 95% of their time idle (Naone 2009). That does not do much to improve power consumption, because even in its idle mode, an idle server still consumes about 70% of peak power (Naone 2009). Power waste on that scale is a major example of energy inefficiency, and idle datacentre servers, if turned off, would reduce the amount of energy consumption.

No standard metric to measure energy efficiency of servers:

If energy efficiency optimisations are to be realised, an energy efficiency metric for servers should be used to ascertain which are most energy-efficient so that scheduling algorithms can be used to decide which server to run. Choosing the best resources in this way will result in the maximum energy efficiency. While there are metrics that focus on IT efficiency

(Blackburn et al. 2010), they do not offer an easy-to-use benchmark to drive energy efficiency optimisation (Whitney and Delforge 2014).

There continues to be low adoption of energy efficient solutions:

The NRDC report already referred to (Whitney and Delforge 2014) says that there are big Cloud farms that perform well in the matter of energy efficiency, but these are responsible for less than 5% of the total energy used globally by data centres. The remaining 95% of data centres, embracing small and medium sized corporate operations and multi-tenant operations are, on average, far less efficient. If energy efficiency best practices were more widely adopted in the small and medium-sized operations that are responsible for the total power consumption of all data centres, there would be a big improvement in overall data centre energy efficiency.

3.2.1.2 Power saving strategies in Cloud

Cloud data centres have three main policies for saving power:

- Dynamic frequency voltage scaling (DVFS);
- Powering down servers; and
- VM consolidation.

Dynamic frequency and voltage scaling (DVFS):

Dynamic voltage frequency scaling (DVFS), also known as CPU throttling, reduces power consumption at times of low loading by dynamic scaling of CPU voltage and frequency. Activation of DVFS can be triggered by a number of policies; Linux kernel, for example, permits DVFS in: Performance, PowerSave, User-Space, Conservative, and OnDemand policies. The governor for each policy decides whether or not to update the frequency (Guérout et al. 2013). There is a price to pay for DVFS. It reduces the number of instructions executed by the processor when a program is run, causing longer run times for

programs and a reduction in performance (Beloglazov et al. 2011). DVFS is also hardware-dependent and not variable as needs change so that, when compared with the other methods, the power savings are low.

It is also a fact that DVFS only acts at server level; even when doing nothing at all, an idle server still consumes up to 70% of power, so that the savings from DVFS are limited. Because of this, other solutions have been developed to consolidate workloads onto a reduced number of servers, and to switch idle hosts off or put them into a lower power consumption mode.

Powering down servers:

Powering down or switching-off servers that are not being used reduces energy consumption. The fact is that many datacentre servers spend most of their time idle and so can be switched off or transferred to sleep mode while they are not being used. Dynamic capacity provisioning or dynamic shutdown problem of this sort must be planned carefully because there are a number of factors to take into account when deciding which servers to power down. Dynamic On/Off programs that turn servers on and off to minimise energy use have been proposed a number of times (Q. Zhang et al. 2012), (Chen et al. 2008), (Guenter, Jain, and Williams 2011), (Kusic et al. 2008) and (ORGERIE and LEFEVRE 2011). Though complex, it is a technique that works well and is capable of reducing power consumption significantly.

Energy-aware consolidation:

Workload consolidation onto fewer servers has become a main power saving technique in Cloud data centres. The object is to select those servers that use energy in the most efficient way and thereby reduce energy consumption (Srikantaiah, Kansal, and Zhao 2008).

Virtual Machines (VM) live migration makes possible Dynamic Optimization and even greater workload consolidation onto an even smaller number of servers and has therefore

come to be seen as a necessary means to move virtual machines between hosts without the need to reboot the VM operating system.

3.2.2 Energy Efficient Cloud Resources Allocation

The cloud infrastructure faces the challenge of efficient allocation of resources as a service provider. Actually, the proposal of the research and academia communities suggests the utilization of diverse resource allocation techniques that help in maintaining the SLA.

3.2.2.1 Resources Allocation

The development of a dynamic method for resource allocation entails the considerations of the SLA between the Software-as-a-Service (SaaS) provider and the user during the resource allocation (Nair and Porwal 2010). The SaaS provider deals with the heterogeneity of the VMs, maps the requests of the customers to parameters of infrastructure levels, and manages the changes in requests of customers. The technique considers the QoS (Quality of Service) for the customer including the response time and the parameters of infrastructure levels. However, the major challenge for the IaaS cloud provider may be in the evaluation of the SLAs between the SaaS providers and the user especially with the large number of SaaS providers. The resource allocation technique that is based on priority is another approach to allocating the resources as presented in (K C Gouda, Radhika T V 2013), (Pawar and Wagh 2012). These approaches are categorized into the resource priority based and the user priority based. Essentially, the approaches have bias considerations for the single service provider that supports in solving the problem of load balancing. The introduction of the resource allocation based on a neural network in (Dinesh, Poornima, and Kiruthika 2012) focuses on the maximization of the use of the resource through the strategy of allocating resources offered by a genetic algorithm. The technique focuses on the resource-abundant systems. In that regard, the users do not compete for the resources.

The cloud resource allocation is the recent focus of the researchers who apply schemes of auctioning to the SaaS providers. Therefore, the requests of the SaaS providers are accepted by the IaaS cloud providers who auction the cloud resources to allocate the highest bidder. The majority of the researchers are focusing on the approaches of game theory studies to

solve the complexities of allocating resources in evolutionary and dynamic environments. The mechanism of allocating the resources based on the game theory has been focused on cloud computing for addressing the problem of optimization of resource allocation (Jebalia et al. 2013). However, according to the recent surveys, the techniques fail to consider the parameters such as resource reliability, execution efficiency, service deadline, resource availability, and fairness. Moreover, a combinatorial auction-based mechanism is investigated for the pricing and allocation of the VMs in the platforms of cloud computing. The approach relies on three schemes that are the greedy scheme, linear programming, and the fixed price scheme. The approach has the weakness of only considering the maximization of the user gains while limiting the allocation of the VM types to a value that is pre-determined (Zaman and Grosu 2013).

3.2.2.2 Cloud Resources Allocation

Resource allocation or scheduling counts among cloud computing's most important tasks. It involves analysing every incoming user request, identifying the resources best suited to meet it, and then assigning those resources in order to meet both the requirements of the user and the cloud provider's goals. Those goals are likely to include energy consumption or cost optimizing. Using information it holds about all resources, together with details of the incoming request and the goals the Cloud provider has, the resource scheduler allocates resources in a way illustrated in Figure 3.6. Schedulers can manage initial and static resource allocation after the arrival of each individual request, or may employ continuous resource management through static and dynamic resource allocation in order to achieve maximum optimisation – if that is the case, then previously received requests may need readjustment from time to time.

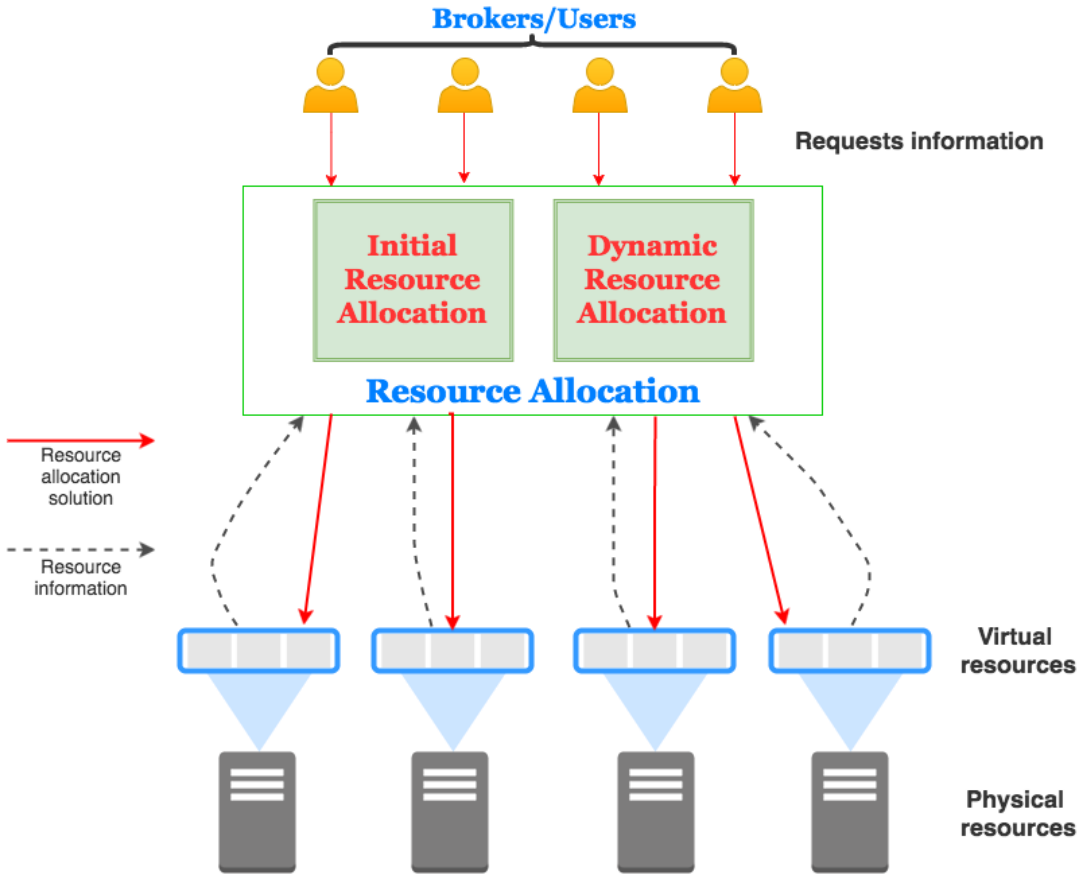

Figure 3.6 Resource Allocation (Nair and Porwal 2010)

As cloud computing and virtualization technologies have become ever more widely adopted, cluster sizes have grown and they may now be in the hundreds or even thousands of nodes for, respectively, small and large data centres. The rise in consumption of electricity resulting from this growth has been enormous and has brought with it an equally enormous rise in the costs of data centre ownership and in data centres' carbon footprints, so that energy efficiency has become a matter of great importance for data centres and Clouds.

A central issue is to optimise resource allocation in Cloud while achieving maximum energy efficiency. This issue is known as NP-hard, which has been extensively studied. The following section will review existing literature concerning energy efficient resource allocation in Cloud.

3.2.2.3 On-demand resource allocation vs advanced resource reservation

The two most significant provisioning plans that cloud providers can offer are: on-demand and reservation plans. On-demand plans means that users can have access to resources when they need them, while reservation plans require the resources to be reserved ahead of time in order to ensure that it is available and free when needed.

On-demand resource allocation:

The majority of Cloud providers allocate resources through such simple methods as immediate on-demand resource allocation, in which resources are allocated if they are available, and if they are not available then the request is scaled out to another provider.

A number of authors (Srikantaiah, Kansal, and Zhao 2008), (Beloglazov and Buyya 2010), (Murtazaev and Oh 2011) and (Li et al. 2009) put forward energy-aware heuristic algorithms and policies with a view to saving energy by reducing to a minimum the number of servers running at any one time. This approach consolidates applications or tasks onto the smallest possible number of servers, with all of the service being switched off. The authors of Chimakurthi and D, (2011) introduced a nature-inspired VM consolidation algorithm that had been inspired by watching Colony Optimization of ants. That algorithm had the same objective: to save energy by reducing the number of machines in use.

The work we've discussed so far studies ways to reduce consumption of energy by cloud data centres using on-demand and immediate algorithms for energy-efficient resource allocation. Theses algorithms are derived for these homogenous data centres with embedded probes and monitoring capabilities such as smart PDUs (power distribution units) or embedded tools for estimating power consumption. Today, most data centres are regarded as very large data centres comprising heterogeneous servers (Koomey 2007) but lacking the ability to monitor energy use. In the next section, we will provide details of algorithms for

on-demand allocation of cloud resources, with division of solutions between static and dynamic.

Advance resource reservation:

What is called “advance resource reservation” simplifies future resource planning and raises the probability that it will be possible to allocate resources as they are demanded. There are many advantages in advance reservation of resources, and it remains the most used, among Cloud providers, on-demand resource allocation approach.

Haizea scheduler (“Haizea - An Open Source VM-Based Lease Manager” 2017) is an open source resource lease manager supporting resource allocation policies of four different kinds: immediate, best-effort, advance reservation (AR) and deadline sensitive. Users ask for AR lease when they need to ensure the availability of infrastructure at lease start and end times which are fixed. Resource reservation is carried out through a mapping function in which there are two dimensions to a slot table: physical nodes; and duration. Resources requested are mapped to physical servers on the basis of availability revealed by the slot table for a specific time interval. Haizea determines how VMs are mapped to servers by use of a greedy algorithm which sorts servers according to loading from low to high after which it works through the node list trying to map the maximum possible number of lease nodes to each server before moving to the next node. Haizea’s scheduling algorithms are simple and greedy and do not take energy efficiency into account (Sotomayor, Keahey, and Foster 2008).

Nathani, Chaudhary, and Somani, (2012) and Loganathan and Mukherjee, (2013) put forward advance resource reservation algorithms for IaaS (Infrastructure as a Service). These algorithms are queue-based and work by checking whether or not sufficient resources will be available for the time period the user asks for. They are concerned with booking resources, and take no account of energy efficiency.

3.2.2.4 Static vs dynamic Cloud resources allocation

The two types of resource allocation are static and dynamic allocation. The first of these, static resource allocation, takes place at the moment a request arrives, while dynamic resource allocation carries out continuous resource management, optimising and adjusting requests previously acted on as well as new requests. VM live migration manages allocation or consolidation of resource with the aim of reducing to a minimum the number of servers activated and in use.

Initial Cloud resources allocation Energy efficient algorithms:

Among the resource allocation mechanisms currently in operation in Cloud data centres are load balancing, round robin and greedy algorithms. OpenNebula ("OpenNebula" 2017), Eucalyptus ("Eucalyptus - Private / Hybrid Cloud Solution" 2017) and OpenStack (OpenStack 2017) Cloud managers use algorithms that are greedy or round robin based and take no account of energy efficiency.

In T. V. Do, (2011), the authors suggest a simple form of energy-aware policy that would incorporate schemes allocating virtual servers to provide green computing. The allocation schemes considered include round-robin and first fit, and reduce energy consumption by setting to a state of reduced power consumption servers that are not hosting VMs. Under this policy, idle servers would be set to a state of low energy consumption, and returned to a fully functioning operating state when in use. A number of authors, (Mazzucco, Dyachuk, and Deters 2010), (Tien Van Do and Krieger 2009), (Mitrani 2011), (Mitrani 2013) and (Tien Van Do and Rotter 2012) all put forward policies that would provide dynamic on/off based on queuing models and heuristic-based methods. Von Laszewski et al., (2009) suggest an approach that would schedule virtual machines to reduce power consumption through Dynamic Voltage Frequency Scaling (DVFS). The energy efficient algorithms put forward

by others (Beloglazov and Buyya 2010) and (Quan et al. 2011) use consolidation policies that minimise the number of servers in use while still accommodating all requested VMs. Heuristics were proposed in each case to solve the bin packing problem as algorithms for VMs consolidation.

The authors of (Srikantaiah, Kansal, and Zhao 2008) consolidate all applications and tasks onto a smaller number of physical machines in order to be able to power off machines not being used. The heuristic they propose for multidimensional bin packing indicates that reduced energy consumption can come from using fewer physical hosts. In Y. Song et al., (2009), the authors propose a multi-tiered resource scheduling scheme providing on-demand capability to hosted services by way of resources moving between VMs and introducing a global resource flowing algorithm to produce the optimum allocation of resources among applications. Since both of these approaches operate at the task level, they would constitute a good fit for Platform as a Service (PaaS) and Software as a Service (SaaS). Allocation is static.

VMs migration Energy efficient algorithms:

Other authors (Issarny, Schantz, and Neogi 2008) put forward a power-aware server consolidation framework, which they call pMapper. It optimises VM placement continuously in order to reduce power consumption to a minimum, relying on greedy heuristics for bin packing problem and introducing VM migration cost, but with no indication about how this is calculated. Hermenier et al., (2009)] propose a similar framework, in this case called Entropy, which acts as a resource manager for homogeneous clusters and carries out dynamic consolidation based on constraint programming while taking account of the migration overhead.

Policies for dynamic VMs reallocation using VMs migration according to CPU performance requirements are put forward in Beloglazov and Buyya, (2010). The most effective of these

ideas is a double threshold policy based on the setting of maximum and minimum use thresholds for hosts and keeping between those extremes total CPU utilisation by all the VMs. When a host's CPU utilization exceeds the top limit, VMs are migrated to its back in order; where the lower threshold is exceeded, all hosted VMs would migrate.

Ferreto et al., (2011) deal with consolidating VMs in a server by migrating VMs that have capacity needs that are both stable and steady. An exact formulation was proposed on the basis of a linear program that was described by too small a number of valid inequalities so that problems that involve the allocation to a number of bins or servers a large number of items or VMs cannot be solved. To find a way around this and propose solutions for large sizes, a heuristic was proposed that would use a static and a dynamic consolidation of VMs in order to reduce energy consumption by hosting nodes and servers.

Other authors (Murtazaev and Oh 2011) proposed a server consolidation (Sercon) algorithm that works by reducing to a minimum the number of nodes used in a data centre and also reducing to a minimum the number of simultaneous migrations. Comparing this algorithm with the heuristic FFD (First-Fit Decreasing) Coffman et al., (1999) used for solving the Bin-Packing problem showed Sercon to be efficient at consolidating VMs and minimising migrations, but the fact is that Sercon is not always able to find the optimum solution.

The authors of Li et al., (2009) outlined what they called EnaCloud to execute dynamic live placement in a cloud platform while taking due account of energy efficiency. Their proposal was for an energy-aware heuristic algorithm that would save energy by reducing to a minimum the number of servers running. Chimakurthi and D, (2011) presented another dynamic resource allocation study, in which the VM consolidation algorithm was based on observations of optimisation in an ant colony and aims to save energy by reducing the number of physical machines in use.

Other authors (Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu 2011) put forward two heuristic algorithms for energy-aware virtual machine scheduling and consolidation. These algorithms were based in the one case on a dynamic round-robin approach (DRR) and in the other on a hybrid approach combining DRR and First-Fit. A different VM consolidation power-saving method for data centres relying on the First-Fit bin packing heuristic was suggested in Takeda and Takemura, (2010). In this approach, VMs would migrate on the basis of server ranks, where "rank" is the selection priority the server has, is unique, and is assigned to each server.

3.2.3 Energy efficient service composition

There was a good deal of research into improving energy efficiency even before the cloud began to become important. At that time, the attention was on reducing the energy consumed by computing devices, especially battery driven devices such as laptops and mobile phones, so that their batteries would last longer (Lecue and Mehandjiev 2011; Taleb et al. 2015), as well as making CPUs, drives and monitors more energy-efficient. These measures transferred to the cloud, but the cloud presents greater challenges because of the enormous number of servers and the fact that a cloud datacentre must provide immediate response to user requests, notwithstanding the wild swings that those requests can represent.

Several strategies have been proposed to reduce the amount of power consumed by cloud system servers. For example, anumber of authors (Y. C. Lee and Zomaya 2012)(Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu 2011)(Liang Luo et al. 2012)(Uchechukwu, Li, and Shen 2012)(Pinheiro et al. 2001) put forward early suggestions concerning datacentre level power management, and those suggestions led to techniques that minimised the consumption of power in computing node clusters supporting multiple applications. Foremost among these techniques was keeping the number of physical nodes at the smallest level needed to handle the workload at that moment, with other nodes being switched off. To

set against this, there is a trade-off between power and performance, with a drop in performance and the failure to provide the promised QoS resulting from fluctuations in workload. Two methods were suggested by Ching-Chi et al. (Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu 2011): Dynamic Round-Robin; and Hybrid. These approaches save energy by the scheduling and consolidation of virtual machines. Developing a migration model and a power model made it easier to estimate what power would be consumed for the execution of the workload and through VM migration.

Liang Luo et al., (2012) tracked the relationship between online infrastructure elements and power consumption, and examined ways in which task types could be matched and power of each element adjusted, resulting in an algorithm to govern resource scheduling in the cloud that focuses on optimisation of energy consumption. Yamini and Vetri Selvi, (2010) went the whole hog with a virtualised cloud to resolve problems of energy consumption and global warming, using a small number of servers to provide service for multiple devices.

Service composition automation seeks to resolve situations where the user's needs cannot be met by a single service, and combines a number of services into one large application. Hence, a number of different approaches have been proposed in this domain, including (Lemos, Daniel, and Benatallah 2015)(Cheng et al. 2015)(Garriga et al. 2015). There are several other open source tools for service composition and execution, such as Sword (S. Ponnekanti 2002), ZenFlow (Martínez et al. 2005), and Flow Editor (Pi et al. 2012). It remains true that there has been little work on optimising energy consumption by the compositions that result. The purpose of service composition optimisation is to choose the service components that will meet user needs while providing optimum overall quality. This overall quality embraces a number of metrics including: energy efficiency; performance;

cost; and trust (Yamini and Vetri Selvi 2010)(Lecue and Mehandjiev 2011). Despite this combination, standard algorithms pay relatively little attention toenergy efficiency.

Other authors (Wajid, Marin, and Karageorgos 2013) extend an existing approach (Lecue and Mehandjiev 2011) and analyse its performance examined for optimised service composition and for the streamlining of resource usage which would lead to greater energy efficiency. Both non-functional and functional criteria are used to assemble an array of cloud services in an energy-efficient configuration. Bartalos and Blake, (2012) examine how difficult it is to measure a web service's overall power consumption. Eunjeong Park and Heonshik Shin, (2008) introduced middleware based on SOA for two purposes: to give control over the quality of service provided for mobile applications; and to graph the energy efficiency of a service composition.

Luo, Zhou, and Wu, (2009) chose the composite service that offered best QoS at lowest cost. This approach rests on the Dijkstras search path and assumes an additive quality in such elements of QoS as duration and throughput, but in Hang, Kalia, and Singh, (2012) Hang intimated that it is on the composite service's nature that the additive aspect depends. So, for example, if the services making up a composite service bar are called in parallel, then overall duration is not found by adding individual durations. The approach suggested by Elshaafi and Botvich, (2012) in trustworthiness collaboratively, taking account of practical aspects in determining how trustworthy a component is, based on its environment. Trustworthiness is calculated on the basis of consumer feedback as to a service's reliability and the satisfaction it offers.

Building a service composition that is optimal in QoS terms will frequently be inefficient due to the inclusion of redundant functionalities and/or services. QoS is directly affected by how many services are in a composition. Rodriguez-Mier et al., (2012), for example, suggested the possibility that bringing to a minimum a composition's number of services

will also help to minimise total response time while maximising throughput. X. Wang et al., (2009) suggested minimising the number of services a composition would require during the life of a persistent query through a greedy algorithm that would minimise transmission costs and routing update costs. Rodriguez-Mier, Mucientes, and Lama, (2012) suggest that the biggest disadvantage of methods of service composition currently in use is poor performance for example, when no priority is placed on keeping the number of services to a minimum and the number of services together with their interaction in I/O terms is high.

3.2.3.1. Bin-packing approach

One survey (Wolke et al. 2015) debates on the usefulness of bin packing for dynamic resource allocation in cloud data centres. Another work (W. Song et al. 2014) proposed a practical bin packing resource allocation algorithm that uses virtualization technology to allocate data centre resources dynamically and support green computing by optimizing the number of servers actively used. Cloud management tools such as OpenStack (OpenStack 2017) and Eucalyptus ("Eucalyptus - Private / Hybrid Cloud Solution" 2017) are commonly used in many IaaS cloud environments for resource allocation, and utilize bin packing heuristics for placing incoming VMs on servers (Wolke et al. 2015). In J. Wang et al., (2013), the authors developed a heuristics approach which integrates the classical bin packing algorithm to address the problem of scheduling independent tasks in a computational grid with different priorities and deadline constraints.

Various work existing in the literature consolidated the multi-dimensional bin-packing problem for allocating and minimizing migrating workloads to achieve energy optimal operations. However, there exists no previous work, which uses the bin packing approach for the optimization of the resulting composition in the multi-cloud environment. Given the large number of cloud resources available from multiple clouds, we achieve energy efficiency by using searching and integrating the smallest possible number of services, from

the smallest possible number of service providers and directing the user requests to the data centres based primarily on using the most energy efficient route.

3.3. Discussion and Requirements

Based on the above sections, Finding the most energy-efficient route to the data centre, and best-fit service, in terms of energy consumption to the user is the multi-cloud broker's biggest challenge. Therefore, and as outlined in the first chapter, the aim of this work is to create and evaluate a multi-cloud broker to address this issue as explained in the problem section. This will be based on the development of holistic algorithms that is able to find the most energy efficient route, and integrate the most energy efficient services from the smallest possible number of cloud data centres in order to minimise the data exchange among participating parties.

As stated in the first chapter, there are two main pillars for energy consumption of cloud computing that should be dealt with efficiently and equally to achieve the most energy-efficient cloud computing environment: (i) the amount of energy consumed on transporting the data between the user and the cloud data centre and (ii) the amount of energy consumed at the computation and processing of each service at the data centre.

The cloud network traffic is forecasted to increase about threefold as stated in chapter 1. Routing big data between the cloud computing parties requires a high bandwidth connection, which consumes larger amounts of energy than just processing and storing big data on cloud data centres, and thus, producing high carbon dioxide emissions. When transferring such amounts of data into a data centre located quite far from the users' geographical location, this power consumption becomes significantly high. Hence, it has become a high necessity to locate the lowest energy consumption route between the user and the designated data centre, and services that match the user request, while making sure the users' requirements are met

On other hand, in a multi-cloud environment, where one cloud provider is not enough, service composition is a crucial. Essentially, a user request will traverse the route to service providers, and get a reply traversed back from the providers to the user. This scenario increases in complexity as the number of required providers, increase, to perform users' task that one service provider cannot process alone due to resource limitations, or when only a sub-part of the requested services is available. This is typically manifested in services composition and the broker-based cloud service model that necessitates collaboration among a number of cloud service providers, which formulates what is so-called multi-cloud environment, whether explicitly, or implicitly to yield the service outcomes and end results to the user. We assume that web services from separate businesses coordinate their activities in such a way that any conflict (such as sharable resources, order dependencies, or communication delays) is avoided

As the number of cloud providers and services increase, the composition of many services from different providers becomes a more complicated task in a real multi-cloud environment. This would require a massive amount of data interchange among all service participants and will consequently lead to high levels of energy consumption. The brokers and service providers tend to priorities QoS metrics, such as service security, availability, response time, as these factors attract clients. The communication cost, and sending and receiving data among the composite Web services from different cloud providers can be expensive, and time and energy consuming. Finding the required services from the minimum number of cloud service providers is as important as finding the services themselves. However, what continues to be a challenging and an under-investigated issue is to find the most energy efficient service composition plan, which should have the least possible number of composite services from a minimum number of cloud service providers

that fulfils the user request. This becomes even more challenging as the number of the cloud service providers increased, which is the main challenge that this work tackles.

3.3.1. Requirements

As such, a set of functional requirements is identified as essential for the development of the proposed brokerage approach. These requirements are detailed below and summarised in Table.1, as below:

R1. Standard-based: The proposed broker must take advantage of existing standards and well-established principles such as, Cloud computing principles and existing models of services development. This requirement will enhance reusability, minimise the effort, and facilitate the broker integration with different target contexts. It is also important that the proposed broker complies with the network topologies and standards to allow it to be easily integrated with the existing networks using R.2.

R2. Energy efficient routing to act as an intermediary bridge for directing the user requests to the green data centres (ranked via R3) based primarily on using the most energy efficient route.

R3. Ranking algorithm to rank cloud service providers based on the total energy consumption. The algorithm sorts the cloud data centres in ascending order based on their energy consumption requirements. This way allows examining the data centre that consumes least energy first.

R4. Atomic service algorithm to enable the broker to search for the best possible atomic services, in terms of matching the incoming request and energy saving target. .

R5. Predefined composition plan algorithm to allow the broker to search all predefined composition plans created by service providers, and check their energy consumption in order to find the best possible match for user requirements.

R6. Optimal composition plan to allow the broker to build a new optimal composition plan by selecting the most energy efficient services (based on R4 and R5) from the smallest possible number of energy efficient providers using R3 results.

Table 3.1 Proposed Broker Requirements

Number	Requirements	Reference
1	**Standard-based**	R.1
2	**Energy efficient routing**	R.2
3	**Ranking algorithm**	R.3
4	**Combination algorithm**	R.4
5	**Energy consumption predefined composition plan**	R.5
6	**Optimal composition plan**	R.6

3.5. Summary

To conclude, while many aspects of delivering web services through the cloud are very well developed, there is still very little research into composing such services in an energy-efficient way. So far as we have been able to establish, no previous research into prioritising energy efficiency as the key metric when optimising composition has been carried out. Considering how large the number of available cloud resources is, energy efficiency is achieved by bringing together the most energy-efficient services from the smallest possible number of providers that will meet requirements of the user.

This chapter discussed network routing and presented the routing problem in cloud computing when selecting a service. Some techniques were presented such as segment routing and sensor networks routing algorithms. Moreover, it introduced the energy efficiency in cloud computing and the energy issue within the cloud and its resources. The chapter concluded with an outline of the main requirements for the new energy efficient brokerage solution in multi-cloud environments.

An Energy Efficient Routing Algorithm

4.1. Introduction

The increasing demand for web services has encouraged service providers and data centres to offer bases in every geographical region, which has resulted in a massive increase in the network traffic and consequent energy consumed by the vast infrastructure. It should be noted that even more energy is consumed at the data transfer between designated server and user than is needed for data processing and data storage produce. The greater the distance that data must be transferred around the world, the greater will be the resulting power consumption. If, for example, a UK-based user is accessing a Google datacentre in Hong Kong, simply transferring the data will have a significant impact on power consumption. A higher carbon footprint (Aldawsari, Baker, and England 2015) is also created by the high network speeds and bandwidth needed to accommodate the amount of network traffic and to speed up the process of data transformation. Thus, the above results in a huge challenge to:

- achieve the environmental requirements as published in the 2011 report of PBL Netherlands Environmental Assessment Agency and JRC European Commission; (Jos G.J. Olivier, Greet Janssens-Maenhout, Jeroen A.H.W. Peters 2011) and
- reduce energy consumption (Baer 2008) and lower the volume of CO_2 emissions by 15%-30% before 2020 to keep increases in global temperature below 2^oC.

Energy consumption and CO_2 emissions from cloud computing are, therefore, an environmental concern.

This chapter presents an algorithm aimed at finding the most energy efficient route between the user and cloud datacentre in order to achieve the full green cloud computing.

4.2 Energy Efficient Routing

The model suggested in this chapter is designed to find the route to the datacentre in a multi-cloud environment that gives the best combination of energy efficiency and QoS (Quality of Service). This objective is chosen to reduce the level of energy consumption in broker-based

systems while providing high QoS in accordance with the Service Level Agreement (SLA) set with the user. However, the route to the selected cloud data centre should be chosen carefully as exchanging data with the cloud data centre may consume more energy than processing the service/job in the data centre itself. Hence, there is a need for a cloud computing route selection approach. Energy consumption by data centres has been the subject of a great deal of study, but the study of energy consumption by a cloud computing network has not had much attention. To achieve the most energy-efficient cloud computing environments, attention must be paid to following factors:

1. Service energy consumption within the data centre; and
2. Energy consumption involved in data transfer between user and the cloud.

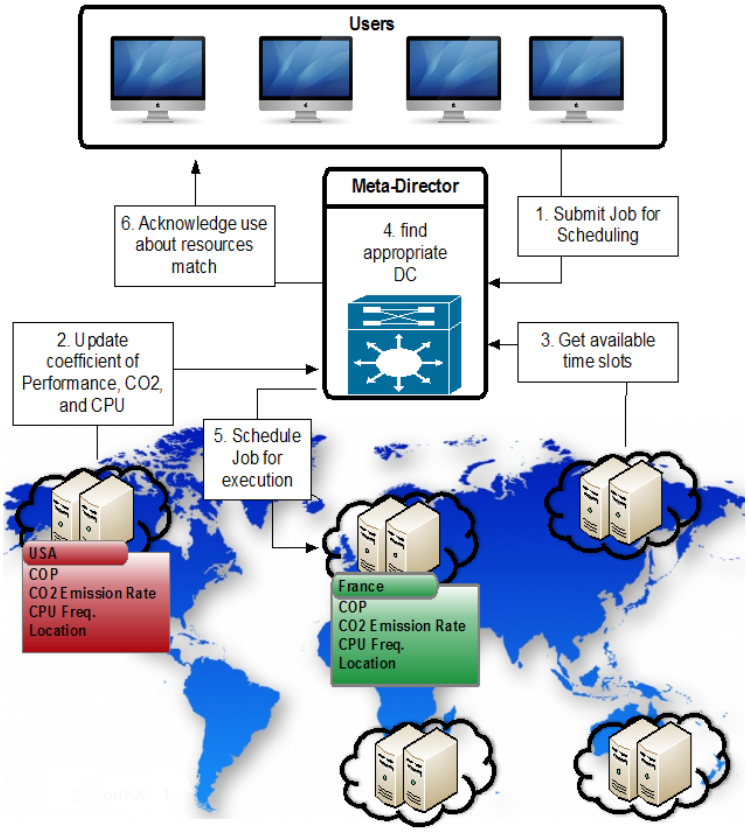

Figure 4.7 cloud elements, that contribute to the total energy consumption.

Most of the work that is being done on cloud computing energy consumption focuses on energy consumed by data centres IT equipment (e.g., servers) and non-IT equipment (e.g.,

light bulbs), without considering the energy consumption by the (alternative) routes to the data centres. Hence, the purpose of this chapter is to develop and evaluate a routing algorithm to bridge the above-mentioned gap. The new algorithm should be capable of routing user requests to the most energy-efficient data centres by selecting, from among various possible routes, the one that is most energy efficient, so that green environment ambitions can be achieved while meeting such user requirements as response time. For this purpose, it is necessary to model the cloud network, taking power consumption into account, in order to establish a workable algorithm. As the network is highly distributed and does not necessarily possess global knowledge of its own state, it is thus necessary to apply a formalism to define the logical state of the user's network that does not rely on explicit state enumeration. For this a situation calculus model will then be used to define the network's logical state in order to confirm the cloud user to green data centre connection. When the connection has been formally established, we calculate both time taken and energy consumed for data transfer and computation. Thereafter, an integer linear programming technique will then be used to model the algorithm. Figure 4.7 shows various elements that contribute to the total energy consumption in cloud environments.

4.2.1 Basics and Rules

As per the literature, there has been a massive amount of research effort in the area of building, or achieving the energy efficient cloud data centres, by reducing the energy required by the data centre infrastructure and/or other data centre necessary equipment. The previous studies in this domain have helped in making the following assumption in connection with a proposed brokerage system:

There are n "green" data centres to which a user machine can be connected through the Internet, to accomplish a certain task.

An available green data centre(s) will therefore be used, and must be capable of being accessed by the route that is selected as the most energy-efficient. Simply put, there are many routes to a green data centre, and the one chosen by the proposed algorithm must be the most energy-efficient among them all.

4.2.2 Modelling power consumption of the network

Understanding the power consumption of the cloud network is deemed essential for modelling the proposed algorithm. A widely accepted way of modelling power consumption by massively distributed infrastructures is based on figures for equipment inventory in telecommunications together with historical sales figures, allowing calculation of energy consumption from knowledge of what type of equipment is in the network, and how many of them are available. This on its own, though, is not enough to determine the actual network architecture and structure. Therefore, to identify the required components and calculate the energy consumption, the network architecture should be known.

A network-based telecommunications model is also required. The network is segmented into: access; metro/edge; core; network; and data centre. Figure 4.8 shows first-cut of a massively distributed network model, and therefore lacks a great deal of the detail of a network's topology and structure, though, it is useful for showing the outline or skeleton of the network architecture as well as the required components in order to calculate energy consumption via using data obtained from the manufacturers on energy consumption of used components. Combining the two approaches (telecommunications equipment inventory statistics based approach and network-based telecommunications) as outlined above, the entire network's power consumption can be calculated using actual infrastructure components. Furthermore, this model also enables power consumption growth to be predicted.

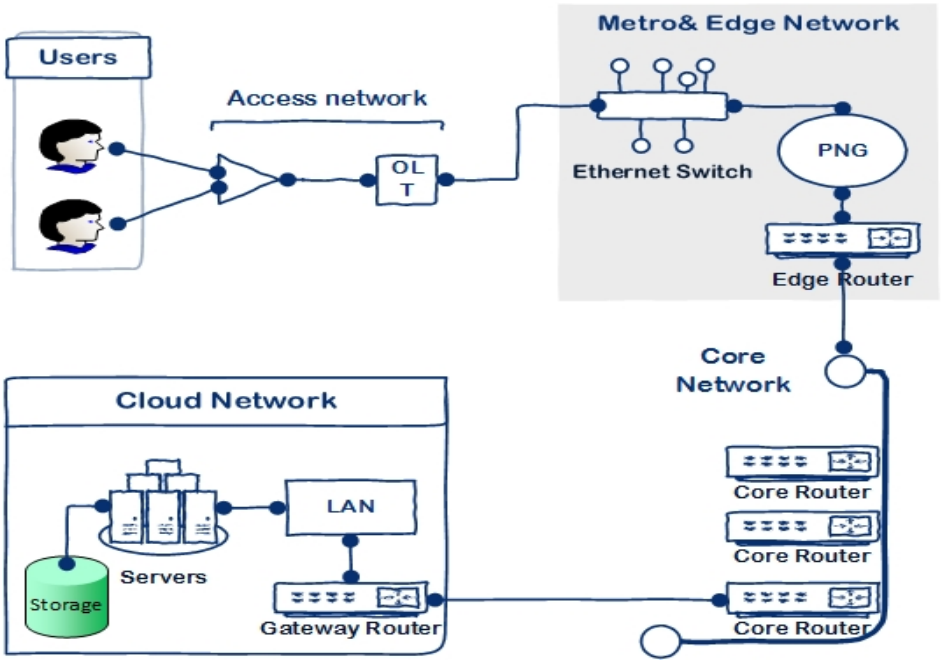

Figure 4.8 Network structure (user connectivity to data centre)

4.2.3 Modelling user connectivity to data centre

The interconnection between a user machine i and a data centre DC_i, via the proposed algorithm, is based on the public cloud structure shown in Figure 4.8 above, which will be formalised as a graph. Thus, between any i and a DC_i, we assume that we have an interconnection graph $G^i = (V^i, T^i, P^i, C^i, E^i, L^i, B^i)$

Where V^i gives a list of all possible nodes available between any i and DC_i; and $Ti : V^i \rightarrow \{1, \ldots, 6\}$ states the nodes' types, which are six different types of nodes available. Therefore, as shown in Figure 4.8, each node v, where $v \epsilon V^i$, might be: an ethernet switch $(T(v) = 0)$, a broadband gateway router $(T(v) = 1)$, a data centre gateway router $(T(v) = 2)$, a provider edge router $(T(v) = 3)$, a core router $(T(v) = 4)$, and a high capacity Wavelength Division Multiplexed (WDM) transport equipment/links $(T(v) = 5)$, to interconnect the core routers, part of the public Internet.

$P^i(v)$ and $C^i(v)$ states the power consumption and the capacity of a node $v \in V^i$, respectively. $E^i \subseteq V^i \times V^i$ defines the interconnection nodes; $L^i : E^i \to \mathbb{N}$ gives the latency between connected nodes E^i ; and finally B^i denotes to the bandwidth.

4.2.4 Formal analysis of network topology

As at first conceived, the network is massively distributed and may not be in possession of a global understanding of its own condition, so a formalistic approach is needed to define the users' network and its logical state without explicit enumeration of the state. To enable this to happen, we propose a calculus of situations as in (Levesque, Pirri, and Reiter 1998), which takes the history of previous *actions* to be a *situation.* In other words, situation (s) is transformed to another situation through the application of action (a) to situation (s), such that *do (a, s)* represents this transformation. The system's logical state is then determined from the initial conditions, effect axioms, frame axioms and qualification axioms. As Levesque, Pirri, and Reiter, (1998) explain, it is often possible to deal with the frame problem through a combination of frame and effect axioms into successor state axioms. In this way, the fitness of a node $v \in V^i$ in a user i's network may be given by:

$$Fitness\ (v, do(a, s))$$
$$= F \Leftrightarrow [\, fitness(v, s) = F \wedge a \notin A(v)]$$
$$\vee \exists m[\, fitness(v, s) = F - m \wedge value(a, s) = m]$$

where $A(v)$ is the action set of node v and value is a function from the set of actions to the integers, mapping each action to a reward (positive integer), cost (negative integer) or no effect (0). This approach makes it possible to define a fairly simple value system so that the choice of components can be optimised from discovery of the datacentre network that is most energy-efficient for this user. As an example, a situation term is added to a node's capacity and power measures:

$$value(addComponent(v), s) = r \equiv P^i(v, s)$$
$$> P^i(v, do(addComponent(v), s) \wedge r = 50 \vee C^i(v, s)$$
$$< C^i(v, do(addComponent(v), s) \wedge r = 100 \vee [P^i(v, s)$$
$$> P^i(v, do(addComponent(v), s)) \vee (C^i(v, s)$$
$$> C^i(v, do(addComponent(v), s)) \wedge r = -10]$$

Thus, for each node: $fitness(do(a, s)) = fitness(s) + value(a, s)$. In addition, probabilities can be assigned to reflect the likelihood of success for network operations. Thus, it can be stated that for any node, the likelihood of connecting/routing to another node v' is dependent on the fitness of that node:

$$prob(addLink(v'), s) = \frac{f_{v'} d_{v'}}{\sum_{v \in V^i} f_v d_v} \quad (4.1)$$

Alternatively, for any $v \in V^i$ this is the probability that $(v, v') \in E^i$. Thus, in order to maintain connectivity:

$$connections(v, do(a, s)) = N \Leftrightarrow [(connections(v, s) = N)$$
$$\wedge a \neq connection\_transfer] \vee connections(v, s)$$
$$= N - m \wedge \exists v'(failed(v', s) \wedge connections(v', s) = m)$$
$$\times poss(connection_{trans}fer, s) \Rightarrow \exists v' failed(v', s)$$

So, if v is the existing node that acquires the connections of a failed node, and v' is the failed unit and v'' is a newnode created, then:

$$connections(v, do(connection\_transfer, s))$$
$$= connections(v, s) + connections(v', s) - m_{vv'}$$
$$- h_{vv'} connections(v'', do(connection\_transfer, s))$$
$$= 1 - - - [A]$$

where $h_{vv'} = 1$ if v was connected to v', and 0 otherwise, and $m_{vv'}$ is the number of nodes with mutual links to v and v':

$$m_{vv'} = \sum_u h_{vu} h_{v'u}$$

For instance, a neighbour node of any particular node is likely to have greater connectivity than that node (Cohen, Havlin, and ben-Avraham 2002)(Thar Baker et al. 2013). In this way, a strategy of ascertaining a green network overlay can be pursued based on identifying only key nodes in the network and routing over these.

4.2.5 Energy required for transportation

For any user's job to be processed, we assume that we have: the quantity of *Flops* that it requires w_u; the amount of input bits in_u to be processed; the amount of output bits ou_u to be returned.

Therefore, if we need an energy of $ET_{send}(i)$ for sending a bit from the user to the data centre and $ET_{recv}(i)$ for the inverse sending, the total energy transportation cost required for processing J_u is: $in_u . ET_{send}(i) + ou_u . ET_{recv}(i)$. To model $ET_{send}(i)$ *and* $ET_{recv}(i)$, we assume that data sent from a user machine to a data centre is always routed on a path that is based the two points connection (the shortest path). In using the formulas proposed in (Baliga et al. 2011), the energy required for sending one bit from a user to a data centre is:

$$ET_{send}(i) = 6\left(\frac{3P_{es}^i}{C_{es}^i} + \frac{P_{bg}^i}{C_{bg}^i} + \frac{P_g^i}{C_g^i} + \frac{2P_{pe}^i}{C_{pe}^i} + \frac{18P_c^i}{C_c^i} + \frac{4P_w^i}{C_w^i}\right) \quad \textbf{(4.2)}$$

where in this case, $P_{es}^i, P_{bg}^i, P_g^i, P_{pe}^i, P_c^i$ and P_w^i represent the power consumed by the nodes types listed in subsection 4.2.3., Ethernet switches, broadband gateway routers, data centre gateway routers, provider edge routers, core routers, and WDM transport equipment, that are located on the path used for routing a user's job to a DC_i. $C_{es}^i, C_{bg}^i, C_g^i, C_{pe}^i, C_c^i$ and C_w^i are the

capacities of the corresponding equipment in bits per second. The values P^i and C^i depend on the nodes used.

Since the above equation does not take into account the power consumption of the other overheads in the cloud network, hence, the entire equation is multiplied by the left factor (six). The factor of six stands precisely for the power requirements for cloud redundancy (factor of 2), cooling equipment and other overheads (factor of 1.5), and the fact that today's network typically operates at under 50% utilisation while still consuming almost 100% of maximum power (factor of 2). The factor of three for Ethernet switches is to include the Ethernet switches in the metro network as well as the Ethernet switches in the LAN inside the data centre. The factor of two for provider edge routers is to include the edge router in the edge network and the gateway router in the data centre, and in the same vein for the other factors in the equation.

Let's consider that G^i comprises the set of paths $Pth = \{pth_1, \dots pth_l\}$ from a user machine i to the data centre DC_i. Then, if the path pth_p was used for sending data, we will have:

$$p_{es}^i = \Sigma(u,v) \in pth_p | T(v) = 0 \; P^i(v) \quad \text{and}$$
$$c_{es}^i = \Sigma(u,v) \in pth_p | T(v) = 0 \; c^i(v)$$

And in the same vein, for the other nodes' types. For example, for the broadband gateway router, p_{es}^i and C_{es}^i will consecutively be:

$$p_{es}^i = \Sigma(u,v) \in pth_p | T(v) = 1 \; P^i(v) \quad \text{and}$$
$$c_{es}^i = \Sigma(u,v) \in pth_p | T(v) = 1 \; c^i(v)$$

4.2.6 Time required for transportation

We assume a simple communication model, Store and Forward, where each node waits for a complete reception of the data before processing it. The approximate time required for sending α bits on a link $e \in E^i$ is equal to: $\max \{L^i(e), [\frac{\alpha}{B^i_{(e)}}] \, . \, L^i(e)\}$.

where, as mentioned in subsection 4.2.3 above that, $L^i : E^i \rightarrow \mathbb{N}$ gives the latency between connected nodes $e \in E^i$; and B^i denotes to the bandwidth. The idea behind it is that either, the bandwidth can contain the bits to send or, we must divide the data to send it in various blocks based on the bandwidth. Finally, we assume that the paths pth_p and $pth_{p'} \in Pth$ were used for sending user data in both directions; then, the total time required for the transportation of a Job J_u in both directions is equal to:

$$Tr(u,i) = \sum_{e \in pth_p} \max \{L^i(e), [\frac{in_u}{B^i_{(e)}}] \, . \, L^i(e)\} + \sum_{e \in pth_{p'}} \max \{L^i(e), [\frac{ou_u}{B^i_{(e)}}] \, . \, L^i(e)\} \tag{4.3}$$

4.2.7 Energy and time required for computation

We assume that each job J_u will be processed by a single machine in the data centre. We also assume that each data centre DC_i is made of a finite set of homogeneous machines that consume $EP(i)$ for processing one flop. Therefore, for processing a job J_u, the data centre DC_i will consume $w_u : EP(i)$. Finally, any machine in a data centre DC_i needs approximatively $\mu(i)$ time units for processing one flop. The job J_u can then be processed in approximatively $w_u . \mu(i)$ times units.

4.3. Implementation

4.3.1. Linear programming formulation

The proposed Green Director (GreeDi) algorithm will be used to direct users' jobs. It routes users' jobs to subscribed green data centres by the route that is most energy-efficient, so that both energy consumption and Service Response Time (SRT) are minimised. This gives rise

to a computational problem requiring resolution: m users' jobs $J_1, \ldots, J_m$ have been submitted to the framework gateway, which is shown here as a server connected to each data centre DC_i by an interconnection graph G^i. On submission, each user's job is accompanied by an *intention* file providing such non-functional SLA requirements such as the maximum response time expected by the user for the processing of that job. A capacity q_i stating the maximal number of jobs that GreeDi algorithm can route on it is associated with each data centre DC_i. Negotiations between framework and cloud provider set this parameter. q_i is also important in ensuring that the response time for dealing with users' requests is minimal. For each job J_u , a data centre must be chosen by the gateway in such a way that total energy consumption both for data transfer and for processing is minimised, while data is processed within the minimal response time contained in the intention file for that user. This is a linear programming formulation in which the decisional variable $x(i,u) \in 0,1$ defines whether or not data centre DC_i will process job J_u. If the Maximal Service Response Time for job J_u (as defined in user's intention file) is $MSRT_{u,}$ then the following mixed integer linear programme is a way of representing the problem:

Model LP_1 :

Minimise $Z = \sum_{u=1}^{m} \sum_{i=1}^{n} x(i,u).[w_u . EP(i) + in_u . ET_{send}(i) + ou_u . ET_{Recv}(i)]$

Subject to:

1. $\forall J_u,\ DC_i : x(i,u) \in \{0,1\}$
2. $\forall J_u : \Sigma_{i=1}^{n} x(i,u) = 1$
3. $\forall J_u : \Sigma_{i=1}^{n} x(i,u) \cdot [w_u \cdot \mu(i) + Tr(u,i)] \leq MSRT_u$
4. $\forall DC_i : \Sigma_{u=1}^{m} x(i,u) \leq q_i$

Any LP_1 solution states to route job J_u towards the data centre $DC_i \; if \; x(i,u) = 1$. In this model, constraint 3 is set to the maximal response time that users expect. It is only possible to guarantee this maximum if the maximal number of jobs allowed to be processed in

parallel at any data centre is limited, and therefore we have constraint 4, where the maximal number of jobs is denoted by q_i.

We assumed that in the case of LP_1 that two paths, pth_p and $pth_{p'}$, would be used to send user data in both directions. Different values for Z might be produced by different path selection. There are two options that allow this inclusion:

- Include it in LP_1, in which case it would be difficult to avoid non-linear equations; or
- Execute the linear program a number of times with different path choices each time until the program returns the answer needing to minimise Z.

This approach is more efficient since we remain with a linear model. Algorithm 1 below summarises LP_1.

Algorithm 4.1. Energy Efficiency Algorithm

Algorithm 4.1 $\boldsymbol{LP_1}$ Input, Output, Steps
INPUT: Jobs $\boldsymbol{J_1, \ldots, J_m}$ with workloads, inputs and outputs data, and intention files; Data centres $\boldsymbol{DC_1}$, … , $\boldsymbol{DC_n}$ with energy consumption per flop and frequency; Interconnection graphs $\boldsymbol{G^1 \ldots G^n}$
OUPUT: Return the best solution on $\boldsymbol{Z}$
STEPS: 1. Define, for each , a set of paths C$\boldsymbol{pth_i}$ that can be used for sending and receiving data. 2. For each *i*, choose a pair of paths $(\boldsymbol{pth_p}, \boldsymbol{pth_{p'}}) \boldsymbol{\epsilon}\ \boldsymbol{Cpth_i}$ 3. Compute the resulting values of $\boldsymbol{ET_{send}(i)}$ and $\boldsymbol{ET_{recv}(i)}$ (equation 2); 4. For any job $\boldsymbol{J_u}$ and data centre $\boldsymbol{DC_i}$ compute $\boldsymbol{Tr(u;\ i)}$ (equation 3) 5. Run $\boldsymbol{LP_1}$ and obtain $\boldsymbol{Z}$; if it is the best obtained value then it will be kept. 6. If there is possible combination $(\boldsymbol{pth_p}, \boldsymbol{pth_{p'}})$ that has not been explored, go to 2

Where $\boldsymbol{Cpth_i}$ is defined by taking the shortest paths on the bandwidth (a special case), we do not loop in this algorithm. Users' intents for maximal response time may make it impossible to realise LP_1, in which case a goal programming formulation will be used.

4.3.2. Goal programming formulation

For any job J_u, we introduce two real deviation variables d_u^+ and d_u^-. A job J_u can be put on data centre DC_i if:

$$w_u \cdot \mu(i) + Tr(u,i) + d_u^- - d_u^+ = MSRT_u$$

Getting close to user's intents demands that we minimise d_u^+ (the difference between the actual SRT and the user's aspiration). The objective here is to minimise both:

- Deviation from user requirements; and
- Total energy consumption.

To find one function capable of handling both of these objectives involves the assumption that a preference factor β_u exists, that it is defined by the user for each job, and that it indicates the relative importance of minimising SRT over energy consumption. Now it becomes possible to derive the following model LP_2:

Model LP_2

Minimise

$$\sum_{u=1}^{m} (1-\beta_u)\frac{E_u}{E_u + d_u^+} + \beta_u \frac{d_u^+}{E_u + d_u^+}$$

Subject to:

1. $\forall J_u, DC_i : x(i,u) \in \{0,1\}$
2. $\forall J_u, : \sum_{i=1}^{n} x(i,u) = 1$
3. $\forall J_u : d_u^-, d_u^+ \geq 0$
4. $\forall J_u, : \sum_{i=1}^{n} x(i,u) \,.\, w_u \cdot \mu(i) + Tr(u,i) + d_u^- - d_u^+ = MSRT_u$

5. $E_u, : \sum_{i=1}^{n} x(i,u) \; .[w_u \cdot EP(i) + in_u \cdot ET_{Send}(i) + ou_u \cdot ET_{Recv}(i)]$
6. $\forall DC_i, : \sum_{u=1}^{m} x(i,u) \leq q_i$

This modelling embraces goals of two types: user goals, which are submitted by way of an intention derived from an SLA; and minimising energy consumption. In this formulation, we have reduced user intents to a threshold for SRT in accordance with MSR T constraint 4 in LP_2. It would be sensible, however, to consider extending the model so that other requirements can be included; these might, for example, include such things as maximal price and minimal security level for data. It is also important to note that both energy and goal deviation have been normalised to make it possible to compare them, and there is a disadvantage to this in that the objective function becomes non-linear. We will therefore propose the use of dynamic programming in order to compute fast LP_2 solutions.

4.3.3. Dynamic programming approach

In this solution, we maintain a two-dimensional array $\in R^{n \times m}$. Each $Z\ (i,l)$ corresponds to an assignment of the jobs $J_1, \ldots J_{l-1}$ to data centres in which J_l is associated with the data centre DC_i. At the beginning of the algorithm, we compute:

$$Z(i,1) \;=\; (1-\beta_1)\,\frac{E_1(i)}{E_1(i)+d_1^+(i)} + \beta_1 \cdot \frac{d_1^+(i)}{E_1(i)+d_1^+(i)}$$

for any data centre DC_i. Here,

$$E_1(i) = \; w_1 \cdot EP(i) + in_1 \cdot ET_{Send}\,(i) + ou_1 \cdot ET_{Recv}(i)$$

and

$$w_1 \cdot \mu(i) + Tr(1,i)] + d_1^-(i) - \; d_1^+(i) \;=\; MSRT_1$$

For the computation of $Z\ (i,l), l \;>\; 1$, we proceed as follows:

1. We consider the different assignments $Z\ (1, l-\ 1) \ldots Z(n, l-1)$ in which the number of Jobs assigned to DC_i is lower than q_i. We will refer to these assignments as (i,l) compatible ones.

2. If there are no (i, l) compatible assignments, we set $Z\ (i, l)\ = +\infty$
3. Otherwise, we choose the (i, l) compatible assignment with the smallest objective value and sum this value in $Z\ (i, l)$, with the cost required for assigning J_l to the data centre DC_i. In a formal manner, this cost is

$$Z\ (i, l)\ =\ (1 - \beta_l)\ \frac{E_l(i)}{E_l(i) + d_l^+(i)}\ + \beta_l\ \cdot\ \frac{d_l^+(i)}{E_l(i) + d_l^+(i)}$$

At the end, we have the values of $Z\ (i, n)$, computed for each data centre DC_i. We then select the assignment that leads to the smallest objective value. The Bellman rule of this modelling can be resumed as follows: the optimal assignment of Job J_l on the data centre DC_i is obtained from the optimal assignment of Jobs $J_1, \ldots J_{l-1}$ in which the capacity used for the data centre DC_i is lower than q_i. The optimality of this rule can be influenced by the way we sort the jobs. We propose for this to use the user's submission ordering. That is: J_1 is the first submitted Job, J_2 is the second, etc. The advantage of this ordering is that implicitly the first user will have the best services.

4.4. Evaluation

In this section, we set out a scenario designed to illustrate the total energy involved in routing a user request to a subscribed green data centre. First, we present the physical network topology used to evaluate energy efficiency, and then we will indicate the types of nodes by route in the topology and calculate energy consumption to enable the results to be compared.

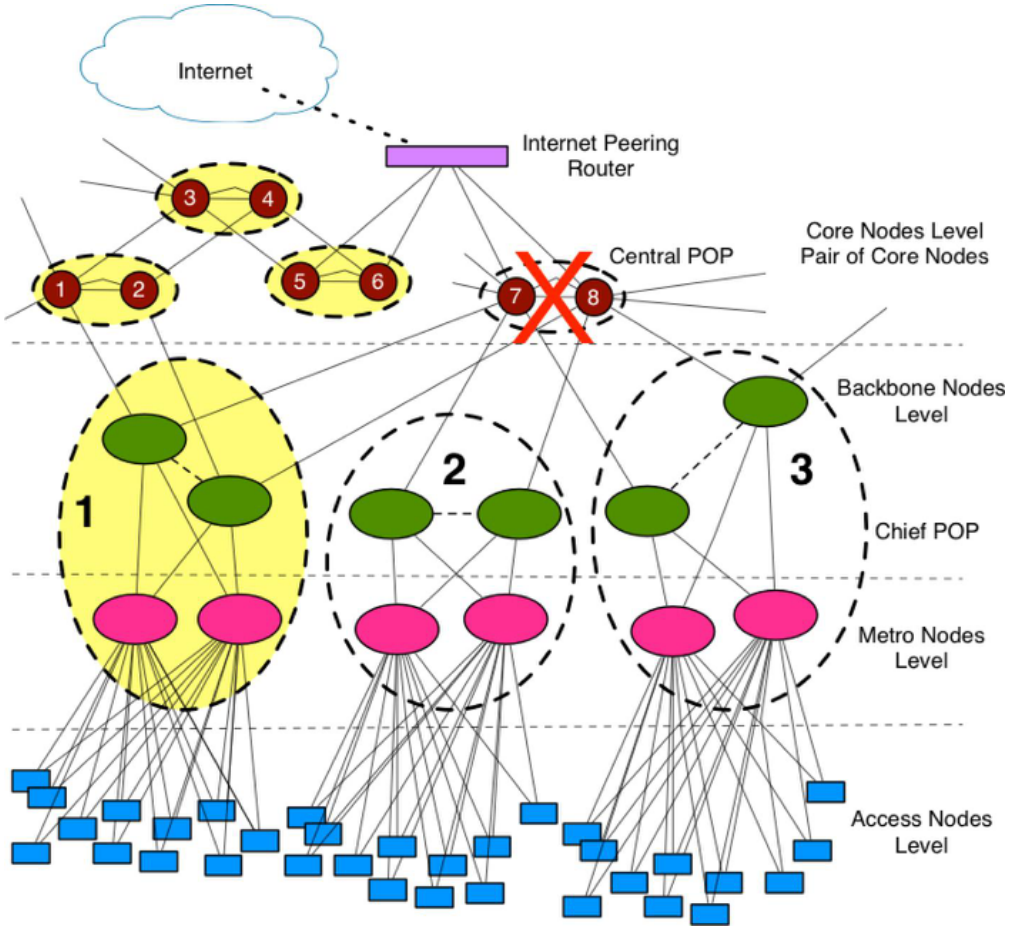

Figure 4.9 Hierarchal topology of an Italian ISP

4.4.1. Physical topology

This network topology makes use of the hierarchal design used by an Italian Internet Service Provider (ISP) (Panarello et al. 2010). This topology has four node levels: core, backbone, metro, and access nodes, with the core nodes as the top level. Central Points-of-Presents (POP's) are mostly to be found in big cities, and this is where the core nodes are located. There is a pair of core nodes in every Central POP, and these core nodes are connected to each other and also to core nodes in adjacent cities. To protect against failure, there will usually be two links for each node-node connection. Internet connection is provided by a high-capacity Internet peering router connected to the core nodes, though it might prove necessary to traverse a number of court nodes before a connection to the Internet is made through a Central POP. The topology's second level comprises the backbone nodes, each of which is connected to two Central POPs.

The physical location of backbone nodes is in the larger POPs, or Chief POPs, distributed among large cities. On the other side from the Central POP connection, the backbone nodes connect to metro nodes, each of which has separate connections to two backbone nodes in case of failure. Location of both Metro and backbone nodes is within the same Chief POP. Access nodes form the lowest level of the topological hierarchy and connect to Digital Subscriber Line Access Multiplexers (DSLAMs). Users connect to these DSLAMs through DSL, FTTN, or PON. The purpose of access nodes is to aggregate traffic from users located within the same area. Each access node is dual-homed to the pair of metro nodes closest to it so that, as Figure 4.9 shows, if any intermediate node should fail, then the user's job will be rerouted to a live node connected to the failed node, depending on the topological level at which failure occurred. It follows that each route's potential power consumption may vary and it is necessary to know energy consumption of each piece of equipment (together with its capacity) on all possible routes beginning from the access node level and continuing all the way to the Internet and then, in a cloud scenario, to the data centre. For this purpose, we will use the power and capacity specification of actual network equipment as given by the equipment manufacturers and apply this information to the topology described. The data in question is shown in Tables 4.3 to 4.6.

Table 4.2: Route A network components

Type	Equipment	Capacity	Power consumption
Ethernet switch (small)	Cisco 4507R-E	64 Gbps	0.658 kW
Ethernet switch	Cisco 6509-E	180 Gbps	2.279 kW
BNG	Juniper E320	320 Gbps	3.347 kW
Provider edge	Cisco 12816	160 Gbps	4.21 kW
Core router	Juniper T640	640 Gbps	6.283 kW
WDM (800 km)	Fujitsu 7700	40 Gbps	136 W/channel

Table 4.3 Route B network components

Type	Equipment	Capacity	Power consumption
Ethernet switch (small)	Cisco 4503	64 Gbps	0.474 kW
Ethernet switch	Cisco 6509	160 Gbps	3.8 kW
BNG	Juniper E120	120 Gbps	1.638 kW
Provider edge	Cisco 12816	160 Gbps	4.21 kW
Core router	Cisco CRS-1	640 Gbps	10.9 kW
WDM (800 km)	Fujitsu 7700	40 Gbps	136 W/channel

Table 4.4: Route C network components

Type	Equipment	Capacity	Power consumption
Ethernet switch (small)	Cisco 4503	64 Gbps	0.474 kW
Ethernet switch	Cisco 6509	160 Gbps	3.8 kW
BNG	Cisco ASR 9001-S	60 Gbps	3.3 kW
Provider edge	Cisco 12816	160 Gbps	4.21 kW
Core router	Cisco CRS-1	640 Gbps	10.9 kW
WDM (800 km)	Fujitsu 7700	40 Gbps	136 W/channel

Table 4.5 Route D network components

Type	Equipment	Capacity	Power consumption
Ethernet switch (small)	Cisco 4503	64 Gbps	0.474 kW
Ethernet switch (Route A)	Cisco 6509-E	180 Gbps	2.279 kW
Ethernet switch (Route B)	Cisco 6509	160 Gbps	3.8 kW
BNG	Juniper E120	120 Gbps	1.638 kW
Provider edge	Cisco 12816	160 Gbps	4.21 kW
Core router	Juniper T640	640 Gbps	6.283 kW
WDM (800 km)	Fujitsu 7700	40 Gbps	136 W/channel

4.4.2. Energy evaluation model and results

The scenario we have chosen has three standard routes to a green cloud datacentre, and each of these routes is structured differently depending on which Chief POP and Central POP are used in the routing, in the matter of the number of nodes traversed, their power, and their capacity. This may be summarised as follows:

1. In Table 4.3, Route A comprises 8 core routers/nodes, 52 edge routers, 52 access routers, 260 residential switches, and 260 end hosts, giving a total count of 632 nodes.

2. In Table 4.4, Route B has fewer intermediate nodes than route A, comprising: 6 core routers, 48 edge routers, 47 access routers, 245 residential switches, and 260 end hosts, for a total count of 606 nodes.
3. In Table 4.5, Route C has the smallest number of intermediate nodes: 5 core routers, 45 edge routers, 45 access routers, 230 residential switches, and 260 end hosts, totalling 585 nodes.

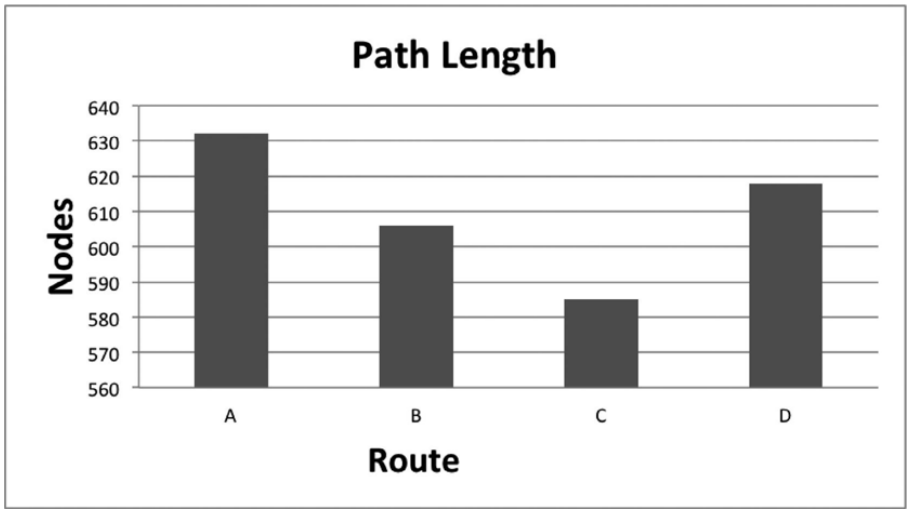

Figure 4.10 Path lengths of each route

Failure of a node can occur at any time while data is being sent and received, and a node failure means that a different route must be chosen and switched to for transmission to be completed. Take the red-crossed node in Figure 4.9, which is a Central POP node forming part of Route A. Should this node experience hardware failure, the backbone nodes will switch to a different Central POP by following the yellow path in the same figure to obtain an Internet connection – and that path is part of Route B. This means that the new yellow route has now become Route D for the purposes of our example and Route D requires three times the number of Central POPs that the original route does. Application of *ETsend* on the new route will show energy consumption might be less than that of the original route, and might be more. Route D (Table 4.6) comprises 8 core routers (all from Route A), 52 edge

routers (all from Route A), 251 residential switches (150 from Route A and 101 from Route B), 47 access routers (all from Route B), and 260 end hosts (all from Route B), giving a total of 618 nodes.

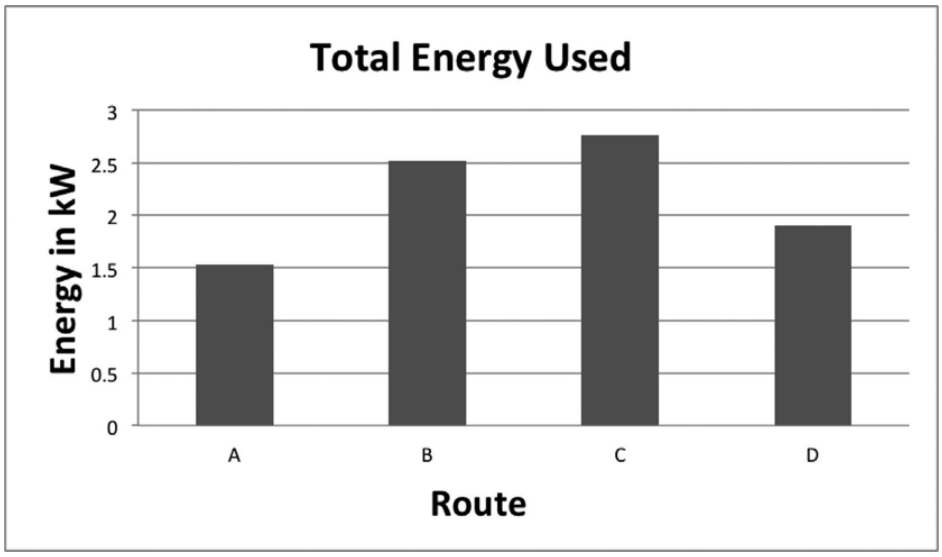

Figure 4.11 Total energy of each route.

Figure 4.10 shows, for each of the four routes we have described, the total number of intermediate nodes on the way to the datacentre. Figure 4.11 and 12 show that the total energy transportation cost for Route A is less than that of any of the other three routes, notwithstanding the fact that Route A covers more nodes than any of the others. The reason lies in the capacity and power consumption of equipment used in the four routes. The values given in Figure 4.11 depend on *ETsend* calculations for the routes, and this is very close to the results from *ETrecv*, where the same route is used for inverse sending. It is therefore clear that the most energy efficient route is Route A, and that – purely from the perspective of energy consumption – this is the most energy-efficient. Route B offers a compromise, combining shortest path possible with energy consumption. Route D may be considered as a recovery route. Figure 4.12 shows results based on the average consumption of energy by each node. Route A, although it is the longest path, has the lowest consumption of energy

per node as well as being overall the lowest consumer of energy. Route B has better overall energy consumption than Route C and is shorter than Route A, but is inefficient in her node energy consumption and its overall energy advantage cannot counterbalance the cost of energy consumed by its additional nodes. It would therefore appear that, from an energy efficiency point of view, the most favourable route is Route A.

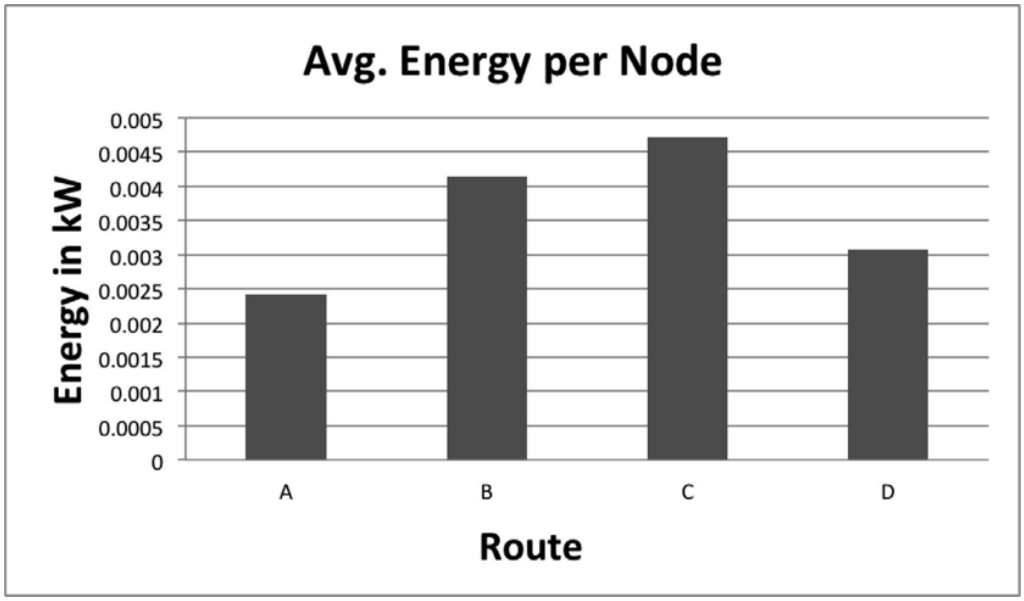

Figure 4.12 Average energy consumption per node of each route.

4.5. Summary

This chapter proposed and evaluated a multi-cloud broker model to act as an intermediary bridge for directing users' requests to the green data centres using the most energy efficient route. In addition, this model aims to achieve the full green cloud computing network ambition. The GreeDi algorithm dealt with the energy efficiency of cloud routing rather than data centres' energy consumption, and proposed and evaluated a new energy efficient routing framework. It was evaluated on a physical Italian ISP topology that has three different routes to a green cloud data centre. From the example results shown in this chapter, the shortest path approach is different from the energy efficient one, and thus, the energy efficient path is used to conform to the environmental objectives.

Bin-Packing Based Energy-Efficient Service Provision

5.1. Introduction

Cloud computing on-demand utility pricing model has grown considerably to support online service delivery, both for businesses and for individuals. The business model is both simple and powerful: providers offer services, end-users find the service they need, and when they have subscribed, high-speed network connection is established between them. In its simplest form, all services required by the user are provided by one cloud provider, but where this is not possible, and more than one cloud is in use, the service must be put together with care. A user request goes to a provider, who returns a reply to the user. When resource limitations or other constraints mean that a single service provider cannot meet all of the user's needs, the model becomes more complex. Typically, this will be seen in composition of services and the use of the model of broker-based cloud services (Aldawsari, Baker, and England 2015), with multiple cloud service providers collaborating in a multi-cloud environment so that the user can receive the desired service outcomes. The assumption must be that the various web services coordinate their activities to avoid conflict in such matters as shareable resources and order dependencies, and also to avoid delays in communication.

The chapter presents a new energy-aware multi-cloud service selection broker, based on the notion of the bin-packing approach, that generates improved composition plans by searching for and integrating the most energy efficient services, or service compositions, from the smallest possible number of competing service providers.

5.1.1. Service Composition Energy Consumption

Cloud computing service composition involves putting together of a set of services that will provide the functionality the user requires. That set of services will be arranged without concern about where the providers' servers are physically located or what level of energy efficiency can be maintained. It is also possible for the service either to be accessed directly

by the user or to be incorporated in other service compositions. Putting together the model of service composition makes use of service-oriented architecture (SOA) and, for this to happen, the following points must be in place:

- Service providers must publish a description of their services including functional and non-functional properties;
- A service composer must construct a suitably able service composition for offer to users; and
- Service discovery must facilitate the matching of user requirements with published descriptions.

Normally, users will pay little attention to where the service is actually being provided from, number of providers involved, or the amount of energy consumed. Rather, their attention will be on speed of the service delivery, response times, and cost.

Research into ways in which energy consumption of the cloud can be reduced is overlooked, whilst finding an energy-efficient cloud computing solution is essential in the pursuit of environmental sustainability. The literature offers quite a number of possible approaches (Luo, Zhou, and Wu 2009)(X. Wang et al. 2009)(Rodriguez-Mier, Mucientes, and Lama 2012) designed to select the optimum mix of service components to combine a suitable environmental approach with such QoS metrics as availability, security, trust, performance, and cost. However, energy efficiency has not been highly prioritised in the process of choosing service components that will offer the most efficiency.

The traditional multi-cloud model involves submitting a request by the user to a service broker setting out the specification of the services required, with the broker then finding a service provider or mix of providers capable of satisfying the request. Finding the best-fit service for the user is, at present, seen as the multi-cloud broker's biggest challenge due to the following reasons:

a) In order to maximise the energy efficiency of cloud services (or resources), it is necessary to find and allocate resources suitable for each incoming request so that the smallest possible number of allocated resources are used to meet user needs, and they are drawn from the smallest possible number of service providers.
b) While a number of heuristic IaaS solutions have been put forward, (Beloglazov, Abawajy, and Buyya 2012)(Kumar et al. 2015) algorithms that will allocate resources in the most energy efficient way are still lacking. As time goes on, there is a development of hybrid cloud solutions combining IaaS and Platform-as-a-Service (PaaS) in a single cloud (OpenStack Heat is an example), and these are attractive because they make it possible for infrastructure and applications to be deployed together, but there is still not sufficient attention to the energy efficient allocation of resources.
c) There is as yet no clear and understood process for finding services that meet user needs and are energy efficient, and nor is it possible for a broker, when drawing together (composing) multiple services, to do so in an energy-efficient way when no single services are available to meet the user request.
d) There is as yet also no way for the broker to compare resources from different clouds in such a way that it would be possible to compose the most energy-efficient plan from the minimum number of clouds.

The steady increase in the number of cloud providers and the number of services makes composing a multi-service plan in a multi-cloud environment ever more difficult. To do so would require that all parties involved exchange data on a huge scale and that, in itself, would increase energy consumption levels. There is a tendency for service providers and brokers to prioritise such QoS metrics as response time, security and availability, because these are the things that attract clients. The cost of transferring data in composite Web

services between more than one cloud provider is high, not only in money terms but also in the time it takes and the energy demands. Optimising required services by locating the smallest possible number of cloud service providers is important, but doing this in the way that consumes the least total energy does not receive enough attention. As the number of cloud service providers increases, so does the difficulty of minimising energy consumption, and that challenge is the central purpose of this study.

5.2. The System Model

In order to formulate the problem and the proposed solution in this thesis, we need to identify the main cooperating parties and their interconnectivity with each other. Fig.13 shows a conceptual representation of the user, broker and cloud service providers. The next subsections formalise the interrelationship among those parties, and show how the new broker works.

5.2.1. Formal datacentre-broker model

When composing a service from a multi-cloud, services requested by the user may be provided by a number of commercial cloud providers. These services are integrated for combined use to communication protocols. The Multiple Cloud service Providers (MCP) is a cloud provider grouping, such that $MCP = \{CP_i, CP_{i+1}, \dots, CP_h\}$ such that $(1 \leq i \leq h)$ represents a CP unique identification number. Because the total energy consumed plays a major role in the algorithm we propose, service providers are expected to provide Total Energy Consumption (TEC) of all services they make available to brokers. The broker then describes each service provider in a 2-tuple $CP_{h,TEC}$. As an example, $CP_{3,174}$ would indicate that all services available from cloud provider 3 had a total power consumption of 174kW. The formulation $\pi_j(CP_{h,TEC_J})$ is used to describe a pre-defined composition plan (j) created by cloud provider (h) with total energy of (TEC).

The algorithm we propose is a benefit of bin-packing and includes, as a constraint, a valid condition. The algorithm's principle is to pack items (which, for this purpose, means finding services) into the smallest possible bin set (the minimum possible number of data centres), as indicated by the data centres' total consumption of power. It is also possible to describe this as packing services requested by the user into the smallest possible number of data centres. In pursuit of this objective, the key decision variable CP_i for each cloud provider i is set to 1 if cloud provider i is selected as a service provider; if not, the value is set to 0. Finding a way to obtain all services requested by the user from the smallest possible number of cloud providers can be expressed as follows:

$$\min I = \sum_{i=1}^{h} CP_i \tag{5.1}$$

The set of web services offered by each cloud service provider is defined as S, where $S(CP_i) = \{S_k, S_{k+1}, S_m\}$ in which $(1 \leq k \leq m)$; k is the unique identification number of each of the m atomic services of CP_i. To enable us to meet the user's request from the smallest possible number of services, another decision variable, s_k, is used, and is set to 1 for a service that has been selected; a value of 0 indicates that the service has not been selected.

$$\min K = \sum_{k=1}^{m} s_k \,|\{\forall\, s : s_{\text{atomic}} \in S \in CP_i\} \tag{5.2}$$

To meet the aims of this research project, the following assumptions underlie the proposed algorithm:

1. Information is provided by every service provider on the total energy consumption of all atomic services at the datacentre (*TEC*), as well as the number of atomic services, the actual pre-defined composition plans $\pi\,(CP)$, and a list of atomic services $\langle s_i;\ EC\rangle$, where *EC* is the energy consumption of service s_i.
2. Cloud providers are listed in ascending order of total energy consumed at that provider by all that provider's atomic services ($lTEC$), so that the first providers to be examined will always be those that consume least energy (Algorithm 5.1).
3. Brokers begin with an examination of atomic services from providers, beginning with the one whose total energy consumption is lowest and ascending in order until finding the first capable of meeting the user's request from a single provider (Algorithm 5.2).
4. If a number of atomic single services are found to meet the user's request, their energy consumption is compared so that the service with minimum power consumption can be selected (Algorithm 5.2).
5. Only if no single provider is found to meet the user's request, then the broker examines predefined composition plans for all providers in ascending order of energy consumption (Algorithm 5.3).
6. If no predefined composition plan that meets the user's request is found, the broker will create one by use of Algorithm 5.4, such that

$$\pi_B = \{\langle s_i, EC(s_i), CP_P\rangle \cup \langle s_j, EC(s_j), CP_q\rangle \cup, \dots, \cup \langle s_k, EC(s_k), CP_r\rangle\}$$

is a set of services either from the same provider, or from a number of providers, subject to:

$$minCons\ \sum_{i=1}^{k}\{EC(s_i)\}\ |\{\forall\ s : s\ \in S\ \in\ CP_r\} \tag{5.3}$$

$$CP_i = \begin{cases} 1, & if\ the\ cloud\ provider\ used; \\ 0, & otherwise. \end{cases} \tag{5.4}$$

$$s_k = \begin{cases} 1, & if\ the\ service\ k\ used \\ 0, & otherwise \end{cases} \tag{5.5}$$

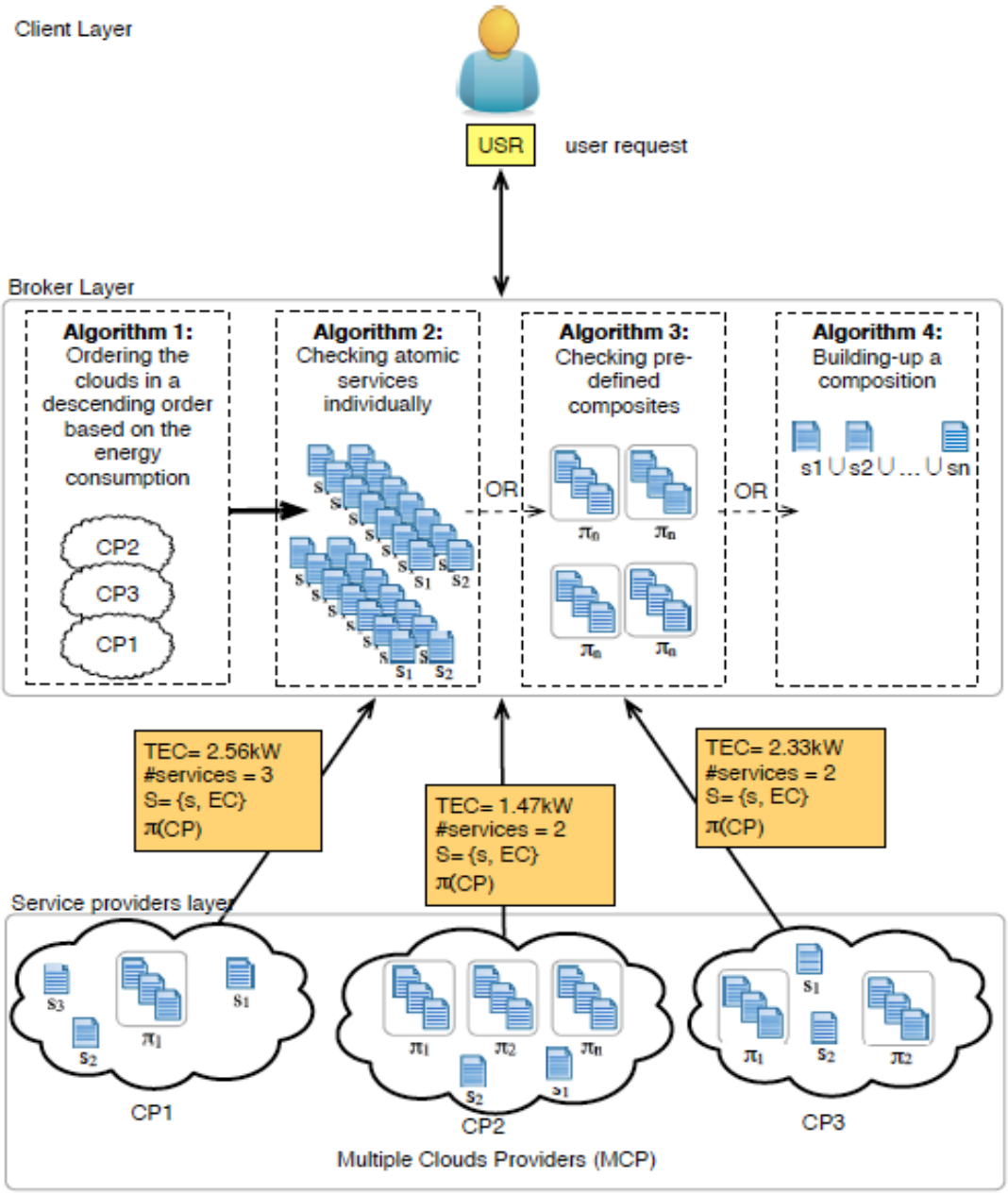

Figure 5.13 Conceptual representation of the proposed approach

Figure 5.13 shows interaction of the four algorithms in the Broker Layer. Algorithm 5.1 is the first step; providers who have subscribed to the broker cloud are reordered on the basis of their total energy consumption. After that, in algorithm 5.2 the broker checks atomic services individually by first cloud provider; if a match is found, the broker replies to the user request to say so, if not the broker will use, Algorithm 5.3 to enable the broker to

examine each provider's predefined compositions to see whether such a composition from a single provider exists to meet the user requirement. Should there still be no match, the broker uses Algorithm 5.4 to build a composition plan that is optimal in terms of energy consumption.

5.2.2. Formal datacentre-broker model

As discussed in (Section 5.2.1), the optimal service or optimal composition plan satisfies the three decision variables in Equations 5.1, 5.2, and 5.3. The optimal plan well, therefore, be:

- An atomic service;
- A predefined composition plan from a subscribed provider; or
- That combination of atomic services, whether from the same provider or from a mix of providers, that combines the lowest possible number of providers with the smallest amount of energy consumed for each selected service.

As an example, take a multi-cloud environment in which the broker negotiates with four cloud providers: $\{CP_1, CP_2, CP_3, CP_4\}$. A set of atomic services is available from each provider, and this set is a subset of the services $\{a, b, c, d, e\}$, and a set of $\pi(CP)$, as shown in Table 5.7.

On receipt of a user request (USR), the broker re-lists the providers in ascending order, producing Table 8 from Algorithm 5.1, and then examines all of the atomic services in row 2 with the aim of finding the most energy efficient to meet the user's needs while consuming the lowest amount of total energy.

Table 5.6 Multiple-cloud providers and Services

Cloud providers	$\langle CP_{4,2.601} \rangle$	$\langle CP_{1,2.35} \rangle$	$\langle CP_{2,1.04} \rangle$	$\langle CP_{3,1.65} \rangle$
Atomic services	a, b, c, e	a, b, c	c, d, e	c, d
EC (kW)	0.52, 0.8, 0.721, 0.56	0.65, 0.5, 1.2	0.72, 0.32	1.2, 0.45
TEC (kW)	2.601	2.35	1.04	1.65
$\boldsymbol{\pi(CP)}$	$\{a, e\}, \{b, c, e\}, \{c, e\}, \{b, e\}$	$\{a, b\}, \{a, c\}, \{b, c\}$	$\{d, e\}$	$\{c, d\}$

It starts first with the service provider that would consume the least total energy, which is $CP_{2,1.04}$ in the case of Table 5.8. For instance, if the user requests services c, then service c from CP_2 will be chosen given that it consumes less energy than any service c from any other service provider

Table 5.7 Multiple-cloud providers and Services sorted by energy consumption

Cloud providers	$\langle CP_{2,1.04}\rangle$	$\langle CP_{3,1.65}\rangle$	$\langle CP_{1,2.35}\rangle$	$\langle CP_{4,2.601}\rangle$
Atomic services	c, d, e	c, d	a, b, c	a, b, c, e
EC (kW)	0.72, 0.32	1.2, 0.45	0.65, 0.5, 1.2	0.52, 0.8, 0.721, 0.56
TEC (kW)	1.04	1.65	2.35	2.601
$\boldsymbol{\pi(CP)}$	$\{d, e\}$	$\{c, d\}$	$\{a, b\}, \{a, c\}, \{b, c\}$	$\{a, e\}, \{b, c, e\}, \{c, e\}, \{b, e\}$

If no match is found, the subscribed service providers' predefined composition plans are tested. If, for example, the user needs services b; c; e, then the order in which the providers will be examined will be: CP_2, CP_3, CP_1, CP_4. The broker checks each composition from each cloud provider to ascertain whether one exists that meets the User Request (USR). If the available matches on more than one, the broker selects the one that consumes the least energy. In this case, since CP_4 has a predefined composition plan that meets user requirements, the need for the smallest possible number of providers in the competition matched with the lowest possible level of energy consumption is satisfied.

5.3. Implementation

5.3.1. Algorithmic Design

There are four main steps by which the broker deals with the request from the user as shown in Figure 5.14. They are based on the multi-cloud environment, subscribed providers, available services and the pre-defined composition plans, where:

a) Step 1, Algorithm 5.1 ranks cloud service providers by total energy consumption.

b) Step 2, Algorithm 5.2 enables the broker to search for the best possible match by seeking a combination of individual atomic service and lowest energy consumption.

c) Step 3, Algorithm 5.3 allows the broker to search predefined composition plans and their energy consumption in order to find the best possible match for user requirements.

d) Step 4, Algorithm 5.4 allows the broker to build a new optimal composition plan by selecting the most energy efficient services from the smallest possible number of providers.

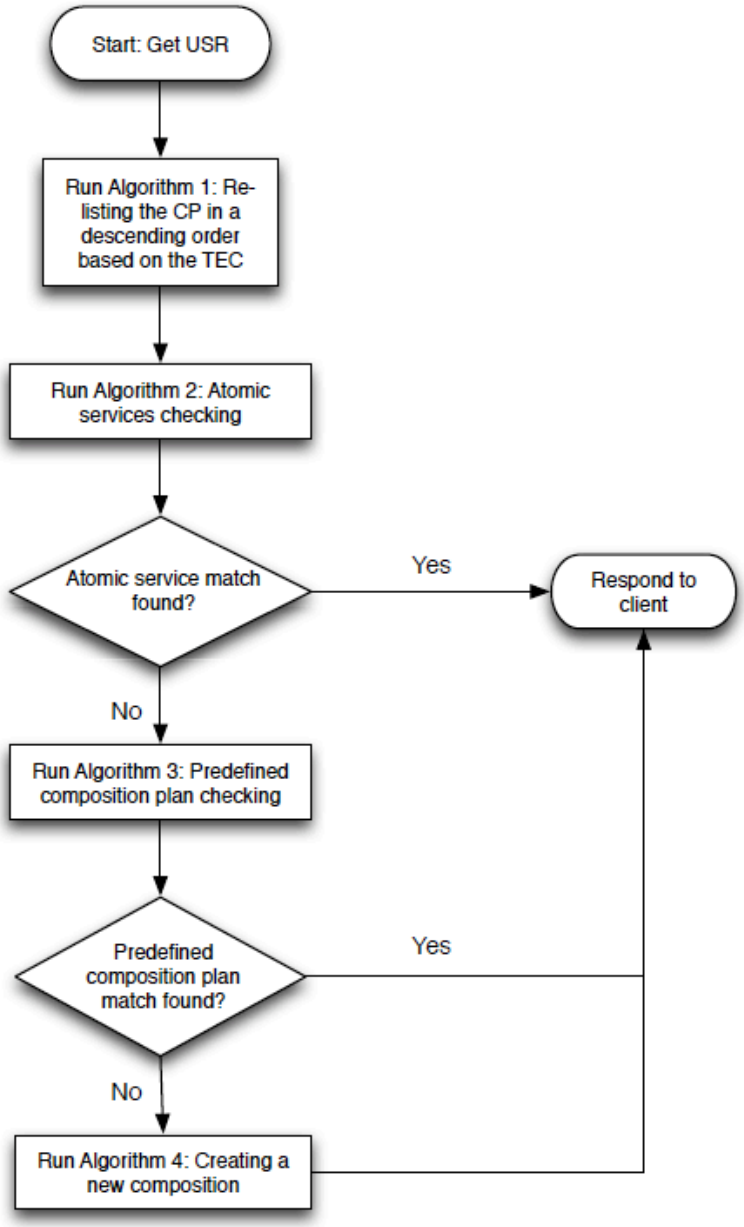

Figure 5.14 Four main steps broker

Algorithm 5.1: Ordering the Cloud Providers in a descending order based on Total Energy Consumption

Algorithm **5.1: Ordering the Cloud Providers in a descending order based on Total Energy Consumption**

input **: number of cloud providers (nCP), Total Energy Consumption** $\{\boldsymbol{TEC(CPi)} \mid \forall \boldsymbol{i} : \boldsymbol{0 < i \leq nCP}\}$

output **: a descending ordered list of cloud providers**

```
1    Get (nCP, {TEC(CPi) | ∀i : 0 < i ≤ nCP}
2    foreach i ←1 to i ≤ nCP− 1 step 1 do
3        lTEC ← i
4        foreach j = i + 1 to ≤ nCP do
5            if EC(j) < EC(i) then
6                ltec ← j
7            end
8        end
9        Temp ← EC(I)
10       EC(I) ← EC(lTEC)
11       EC(lTEC) ← Temp
12   end
```

Algorithm 5.2: Finding an atomic service that matches the user request

Algorithm **5.2: Finding an atomic service that matches the user request**	
Input	user service request (USR), number of multiple cloud providers (nCP)
output	most energy efficient service from most "possible" energy efficient data centre ($\boldsymbol{Ser_k(CP_{i\ lTEC})}$), actual energy consumption of the selected service (minCons)

```
1   Get (USR, nCP )
2   foreach i ←1 to i ≤ nCP step 1 do
3           select (CP_{i,lTEC})
4           Get #(Ser(CP_{i,lTEC}))
5           j ← #(Ser(CP_{i,lTEC}))
6           foreach k ← 1 to k ≤ j step 1 do
7                   if ((Ser_k(CPi lTEC) ∩ USR) == Ø) then
8                           go to 26
9                   else
10                          if (k == 1) then
11                                  minCons ← EC(Ser_k(CP_{i,lTEC}))
12                                  return Ser_k(CP_{i,lTEC}), minCons
13                                  go to 6
14                          else
15                                  if  (EC(Ser_k(CP_{i,lTEC})) < minCons) then
16                                          minCons ← EC(Ser_k(CP_{i,lTEC}))
17                                          return Ser_k(CP_{i,lTEC}), minCons
18                                          go to 26
19                                  else
20                                          go to 26
21                                  end
22                          end
23                          return Ser_k(CP_{i,lTEC}), minCons
24                          go to 6
25                  end
26                  if (k == j)  then
27                          go to 2
28                  else
29                          go to 6
30                  end
31          end
32          if  (i == nCP ) then
33                  go to 37
34          else
35                  go to 2
36          end
37  end
```

Algorithm 5.3: Finding a predefined optimal composition plan π'_B from a single provider

Algorithm 5.3: Finding a predefined optimal composition plan $\boldsymbol{\pi'_B}$ from a single provider	
Input	User service request (USR), multiple cloud providers (MCP), largest number of composition plan m.
Output	Optimal composition plan π'_B
Assumption	Cloud providers are stored in decreasing order based on the number of composition plans
1	Beginalgorithmic [1] USR ← Ø; π'_B NULL; minCons NULL; m largest number of composition plan; ▷Initialise
2	Get USR $\langle I, G \rangle$
3	Select (CP_m) ▷ CP that contains the largest number of composition plans m
4	$i \leftarrow m$
5	if (i is True) then
6	$foreach\ \ j \leftarrow 1$ to $j \leq i\ step\ 1$ do
7	if $(\pi_j(CP_i) \cap USR == \emptyset\ then$
8	if $(j = i)$ then
9	go to 35
10	else
11	go to 6
12	end
13	else
14	if $(j = 1)$ then
15	$minCons \leftarrow EC(\pi_j(CP_i))$
16	else
17	if $(EC(\pi_j(CP_i)) < minCons)$ then
18	$minCons \leftarrow EC(\pi_j(CP_i))$
19	else
20	go to 6
21	end
22	end
23	end
24	end
25	return $minCons$
26	if $(i = m)$ then
27	$\pi'_B \leftarrow minCons$
28	else
29	if ($minCons < \pi'_B$) then
30	$\pi'_B \leftarrow minCons$
31	end
32	end
33	return π'_B ▷ Optimal composition plan
34	end
35	$i = i - 1$
36	if $(i \geq 1)$ then
37	select (CP_i)
38	go to 5 ▷ So that other CPs will be checked in a decreasing order
39	else
40	invoke algorithm 5.2
41	end

Algorithm 5.4: Creating an optimal services composition from multiple providers

Algorithm 5.4: Creating an optimal services composition from multiple providers	
Input **Output**	user service request (USR), number of multiple cloud providers (nCP) most energy efficient service from most "possible" energy efficient data centre (Ser_k (CP_{ilTEC})), actual energy consumption of the selected service (minCons)
1	$serList \leftarrow \emptyset$; $cldList \leftarrow \emptyset$; $minCons \leftarrow \emptyset$; $totalCons \leftarrow \emptyset$
2	Get (USR, nCP)
3	$foreach\ i \leftarrow 1\ to\ i \leq nCP$ Step 1 do
4	Select ($CP_{i,lTEC}$)
5	Get #($Ser(CP_{i,lTEC}$))
6	j ← # ($Ser(CP_{i,lTEC}$))
7	$foreach$ k ←1 to k ≥ $j\ step\ 1\ do$
8	if $\left((Ser_k\ (CP_{i,lTEC}\ )\right) \cap USR) - serList == \emptyset$) then
9	if $\left((Ser_k\ (CP_{i,lTEC}\ )\right) \cap USR) \in serList == true$) then
10	if $\left(EC(Ser_k\ (CP_{i,lTEC}\ )\right) \cap USR) - serList) < EC(Ser_k\ (CP_{i,lTEC}\ )) \in\ serList))\ then$
11	Swap
12	$minCons \leftarrow\ EC((Ser_k\ (CP_{i,lTEC}\ ) \cap USR) - serList)$
13	$totalCons \leftarrow totalCons\ \leftarrow$ minCons
14	go to 7
15	else
16	go to 7
17	end
18	else
19	go o 7
20	end
21	else
22	$minCons \leftarrow\ EC((Ser_k\ (CP_{i,lTEC}\ ) \cap USR) - serList)$
23	$serList\ \leftarrow\leftarrow\ serList\ \cup ((Ser_k\ (CP_{i,lTEC}\ ) \cap USR) - serList)$
24	$totalCons \leftarrow totalCons\ \leftarrow$ minCons
25	go to 7
26	end
27	end
28	if $(i < nCP)$ then
29	$i\ =\ I\ +\ 1$
30	go to 4
31	else
32	go to end
33	end
34	end

5.4. Evaluation

When evaluating the proposed algorithm's performance and efficiency gains that may result from it, there is an advantage in comparing its results against those of established benchmark algorithms so that the potential for reducing energy consumption can be measured. This comparison was carried out using five existing algorithms used in evaluating combinations of cloud services: (All Clouds, Base Cloud, Smart Cloud (Zou et al. 2010), COM2 (Kurdi et al. 2015), and DC-Cloud (Lu et al. 2015)).

5.4.1 Experimental Settings

To accomplish a systematic evaluation, we used the same simulation parameters of the listed methods. The basis of the experimental data was a default web service test set included in the OWL-S XPlan package (Karunamurthy, Khendek, and Glitho 2012). Developing a dedicated simulator allows performance assessments to be conducted and a comparison to be made. Java EE 8 is the programming language used to implement the algorithm, with IBM ILOG CPLEX Optimization Linear Solver ("IBM ILOG CPLEX Optimization Studio" 1AD) acting as the simulated running environment. The hardware was an Apple iMac (Retina display, 2.8 GHz Intel Core i7, and 16 GB 2133 MHz DDR3).

Our simulation starts from an assumption that the broker deals with four cloud providers: CP_1, CP_2, CP_3, and CP_4. Each provider has a set of pre-defined composition plans: $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ based on the Multi-Cloud Providers (MCP) environment in Table 5.13.

Table 5.8 Cloud provider's composition set per MCP.

MCP_S	CP_1	CP_2	CP_3	CP_4
MCP_1	π_1, π_2, π_3	π_4, π_5	π_3, π_4	$\pi_1, \pi_2, \pi_3, \pi_5$
MCP_2	π_1, π_2	π_3	π_2, π_5	π_1, π_4, π_5
MCP_3	π_1, π_3, π_5	π_5	π_1, π_2	π_3, π_4
MCP_4	π_2, π_3, π_5	π_3, π_4	π_1, π_2, π_3	π_4, π_5
MCP_5	π_1, π_2	π_2, π_3	π_3	π_1, π_4, π_5

Table 5.14 shows {2, 3, 8, 3, 3} which represent the number of web services in the composition plans.

Table 5.9 Number of services per composition.

Composition plan	π_1	π_2	π_3	π_4	π_5
Number of services	**2**	**3**	**8**	**3**	**3**

The list of cloud providers will be presented in ascending order of total energy consumption (TEC) for the available set of services, as per proposed Algorithm 5.1 Table 5.15. This order will be different in each MCP for the same provider Table 5.16. Thus, CP_4 is the most efficient provider in MCP_1 and the least efficient in MCP_4.

Table 5.10: CPs and energy consumption per MCPs, before listing in an ascending order.

MCP_S	CP_1	CP_2	CP_3	CP_4
MCP_1	$\langle CP_{1,2.7}\rangle$	$\langle CP_{2,3.1}\rangle$	$\langle CP_{3,3.5}\rangle$	$\langle CP_{4,2.2}\rangle$
MCP_2	$\langle CP_{1,1.9}\rangle$	$\langle CP_{2,2.9}\rangle$	$\langle CP_{3,2.6}\rangle$	$\langle CP_{4,1.7}\rangle$
MCP_3	$\langle CP_{1,2.2}\rangle$	$\langle CP_{2,2.8}\rangle$	$\langle CP_{3,2.4}\rangle$	$\langle CP_{4,2.5}\rangle$
MCP_4	$\langle CP_{1,2.5}\rangle$	$\langle CP_{2,2.8}\rangle$	$\langle CP_{3,2.7}\rangle$	$\langle CP_{4,3.7}\rangle$
MCP_5	$\langle CP_{1,2.4}\rangle$	$\langle CP_{2,2.8}\rangle$	$\langle CP_{3,3.5}\rangle$	$\langle CP_{4,2.2}\rangle$

Table 5.11: CPs and energy consumption per MCPs, after listing in an ascending order.

MCP_S	Sorting order of CPs ascendingly			
MCP_1	$\langle CP_{4,2.2}\rangle$	$\langle CP_{1,2.7}\rangle$	$\langle CP_{2,3.1}\rangle$	$\langle CP_{3,3.5}\rangle$
MCP_2	$\langle CP_{4,1.7}\rangle$	$\langle CP_{1,1.9}\rangle$	$\langle CP_{3,2.6}\rangle$	$\langle CP_{2,2.9}\rangle$
MCP_3	$\langle CP_{1,2.2}\rangle$	$\langle CP_{3,2.4}\rangle$	$\langle CP_{4,2.5}\rangle$	$\langle CP_{2,2.8}\rangle$
MCP_4	$\langle CP_{1,2.5}\rangle$	$\langle CP_{3,2.7}\rangle$	$\langle CP_{2,2.8}\rangle$	$\langle CP_{4,3.7}\rangle$
MCP_5	$\langle CP_{4,2.2}\rangle$	$\langle CP_{1,2.4}\rangle$	$\langle CP_{2,2.8}\rangle$	$\langle CP_{3,3.5}\rangle$

5.4.2 Experimental results

The Results for all five benchmark algorithms, the All Clouds in Table 5.17.a, the Base Cloud in Table 5.17.b, the Smart Cloud in Table 5.17.c, COM2 in Table 5.17.d and DC-Cloud in Table 5.17.e are all consistent with previously published results in (Zou et al. 2010),(Kurdi et al. 2015),(Lu et al. 2015). Table 5.17.f lists the evaluation results for the

new broker so that this broker can be compared with those already discussed. In the first experiment of this chapter the following performance measures were evaluated:

- The number of cloud providers involved in the final composition $|CP|$.
- The number of services checked before reaching the final composition $|S|$.

Table 5.17 shows that our algorithm was successful in improving performance in comparison with other algorithms by keeping the number of examined services and composite clouds low. Services examined $|S|$ did not exceed 38, and the number of combined clouds was as low as two clouds and never exceeded three. Also, of all approaches, ours examined the smallest total number of services (152) and clouds (11), and this directly affects time spent on determining the final composition.

Table 5.12: CPs and number of () composition plans per MCPs

(a) All Clouds Algorithm

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_1, CP_2, CP_4	3	46
MCP_2	CP_1, CP_2, CP_3, CP_4	4	27
MCP_3	CP_1, CP_3, CP_4	3	32
MCP_4	CP_1, CP_2, CP_3, CP_4	4	44
MCP_5	CP_1, CP_2, CP_3, CP_4	4	32
TOTAL		18	181

(b) Based Cloud Algorithm

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_1, CP_2	2	65
MCP_2	CP_1, CP_2, CP_4	3	148
MCP_3	CP_3, CP_4	2	128
MCP_4	CP_2, CP_3	2	68
MCP_5	CP_2, CP_4	2	112
TOTAL		11	521

(c) Smart Cloud Algorithm

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_1, CP_3	2	70
MCP_2	CP_1, CP_2, CP_4	3	48
MCP_3	CP_3, CP_4	2	48
MCP_4	CP_2, CP_3	2	140
MCP_5	CP_1, CP_2, CP_4	3	56
TOTAL		12	362

(d) COM2 Algorithm

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_4, CP_2	2	35
MCP_2	CP_4, CP_2, CP_3	3	45
MCP_3	CP_1, CP_4, CP_3	3	50
MCP_4	CP_1, CP_3, CP_2	3	49
MCP_5	CP_2, CP_4	2	30
TOTAL		14	209

(e) DC-Clouds Algorithm

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_4, CP_2	2	46
MCP_2	CP_4, CP_2, CP_3	3	27
MCP_3	CP_1, CP_4	2	29
MCP_4	CP_1, CP_4	2	44
MCP_5	CP_2, CP_3, CP_4	3	32
TOTAL		12	178

(f) Our Broker

Performance	CP involved	$\|CP\|$	$\|S\|$
MCP_1	CP_4, CP_2	2	35
MCP_2	CP_4, CP_1, CP_2	3	26
MCP_3	CP_1, CP_3	2	29
MCP_4	CP_1, CP_3	2	38
MCP_5	CP_1, CP_3	2	24
TOTAL		11	152

Further validation of our algorithm in terms of gains in time and efficiency came from a second experiment that measured the running time and energy consumption of the five algorithms in order to find the requested composition and then compared those results with the time and energy expended by our broker to produce the same composition. The results,

in terms of actual running time, are shown in Fig 5.15.a. For our broker, results are obtained using a list of clouds pre-sorted in decreasing order of TEC. As Fig 5.15.b shows, this new broker consumed less energy than any of the other approaches.

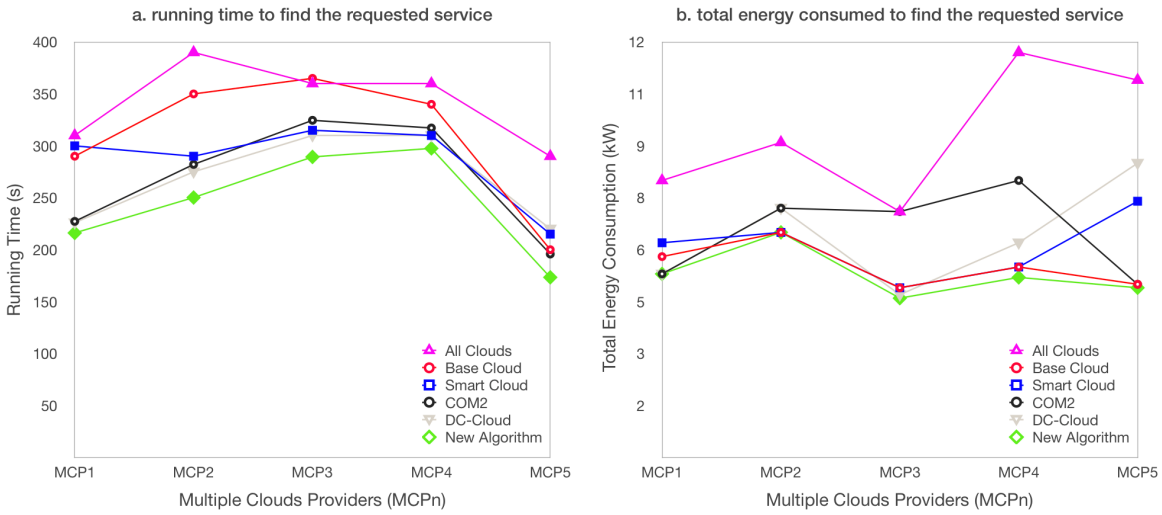

Figure 5.15 Running time and energy consumption to find requested service

5.5. Summary

This chapter presented a novel multi-cloud service computing approach, focusing on the selection of energy-efficient services and service composition plans that meet user requirements. In particular, a Bin-packing based service composition algorithm that determines the minimum number of composite services based on an effective combination of cloud services' providers that satisfy the user needs. Our approach therefore addresses the increasing need for optimising energy consumption associated with the rise in complex (or real-world) cloud based services and user request scenarios, which are characterised by a large number of cloud providers and services. The proposed approach was evaluated against five established service computing algorithms for multiple cloud environments and the simulation results demonstrated that our algorithm produces significant relative performance improvements in terms of both running time and energy consumption.

Energy-Aware Service Composition Algorithm for Multiple Cloud

6.1. Introduction

In many cases, when a single service is not enough to complete the business requirement; a composition of web services is carried out. These composed web services are expected to collaborate towards a common goal with large amounts of data exchange and various other operations. Massive data sets have to be exchanged between several geographically distributed and scattered services. The movement of mass data between services influences the whole application process in terms of energy consumption. One way to significantly reduce this massive data exchange is to use fewer services for a composition, which needs to be created to complete a business requirement. Integrating fewer services can result in a reduction in data interchange, which in return helps in reducing the energy consumption and carbon footprint.

This chapter aims at creating an energy-aware composition plan by searching for and integrating the smallest possible number of services, in order to fulfil user requirements. The algorithm proposed here is different from the one proposed in the fifth chapter as this algorithm checks cloud providers in descending order based on the number of predefined composition plans, while the previous chapter checked in ascending order based on the energy consumption.

6.1.1. Service compositions

In the more conventional multi-cloud environment scenarios, the user submits a request to a service broker declaring the required services specifications. Then the broker needs to find the appropriate service(s) and a service provider who can meet the request. Presently, finding the service that best fits user needs, broker aims, and environmental targets is a challenging task for multi-cloud brokers for the following fact: how might the broker use multiple services to meet the requests, particularly when there is not a single service, which can match the request?

To illustrate further the above issues, the following five different web services with their Web Service Description Language (WSDL) code and a potential composition will be considered throughout this chapter, as shown in Figure 6.16 and Figure 6.17 respectively:

- Given the street address/name as input, geoCoding type service, returns the associated geographical coordinates. In this paper, this is denoted as service a.
- Given the geographical coordinates, pointOfInterest type service returns the places that end users might be interested in. In this paper, this is denoted as service b.
- Given the geographical coordinates, weatherForecast returns the information about the weather observations at the station closest to the end user. In this paper, this is denoted as service c.
- Given the geographical coordinates, map type service returns a map showing the position of the end user. In this paper, this is denoted as service d.
- A webPageInfoCollector type service takes a set of information related to a location as input and returns a web page that shows it. In this paper, this is denoted as service.

Service a	$< xsd: element\ name = 'geoCoding\_Request' >$ $\quad < xsd: element\ name = 'streetName'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$ $< xsd: element\ name = 'geoCoding\_Response' >$ $\quad < xsd: element\ name = 'geoCoordinates'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$
Service b	$< xsd: element\ name = 'pointOfInterest\_Request' >$ $\quad < xsd: element\ name = 'geoCoordinates'\ \ Type = "'xsd: string"/>$ $\quad < xsd: element\ name = 'typeOfInterest'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$ $< xsd: element\ name = 'pointOfInterest\_Response' >$ $\quad < xsd: element\ name = 'namesOfPoints'\ \ Type = "'xsd: string"/>$ $\quad < xsd: element\ name = 'directions'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$
Service c	$< xsd: element\ name = 'weatherForecast\_Request' >$ $\quad < xsd: element\ name = 'geoCoordinates'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$ $< xsd: element\ name = 'weatherForecast\_Response' >$ $\quad < xsd: element\ name = 'temprature'\ \ Type = "'xsd: string"/>$ $\quad < xsd: element\ name = 'windSpreed'\ \ Type = "'xsd: string"/>$ $\quad < xsd: element\ name = 'windDirection'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$

Service d	$< xsd: element\ name = 'map\_Request' >$ $\quad < xsd: element\ name = 'geoCoordinates'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$ $< xsd: element\ name = 'map\_Response' >$ $\quad < xsd: element\ name = 'position'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$
Service e	$< xsd: element\ name = 'webPageInfoCollector\_Request' >$ $\quad < xsd: element\ name = 'location'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$ $< xsd: element\ name = 'webPageInfoCollector\_Response' >$ $\quad < xsd: element\ name = 'webpage'\ \ Type = "'xsd: string"/>$ $< xsd: element/>$

Figure 6.16: Five separate web services.

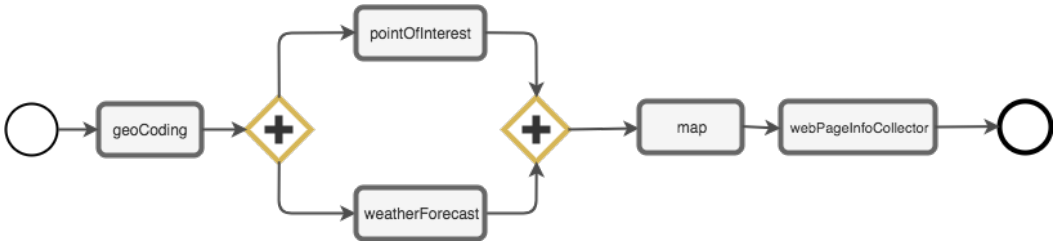

Figure 6.17 : BPMN composition

Now, consider these requests received by the broker:

- **Request 1**: find the Geographical Coordinates of Byrom Street in Liverpool.
- **Request 2**: find the location of Big Ben in the centre of London and the weather forecast in that area, and on the map show directions from the current location to Big Ben.

Request 1 can be fulfilled by invoking service (a); passing the street name to obtain geographical coordinates. But the second request needs services composition, as the request cannot be satisfied by any of the five services individually. Thus, a new service is used, called Information Service (IS), which returns location-based information services (e.g., the current weather or a map showing various Points-of-Interests (PoI) based on the current GPS coordinates of a mobile device, or after an address is entered). Each of the services in the IS

composition is linked to a real remote web service which runs in the background and is registered with a marketplace. Once the street address of the user is provided as input, the composite service returns a web page with user location-based information. Therefore, the second request can be reached by two methods:

a) Invoking the pointOfInterest service, pass the geoCoordinates and the pointOfInterest as an input, to get the directions to Big Ben; then invoke the weatherForecast to find the temperature and other weather activity.
b) Compose the weatherForecast and map web services to fulfil the same request.

The example above clearly shows the composition problem in a "web services composition" domain. Nonetheless, as the number of services and service providers increases, different providers' services become increasingly complicated in a real multi-cloud environment. This needs large amounts of data exchanged between all service participants, which subsequently leads to a higher level of energy consumption.

As mentioned above, usually a cloud user submits a job as a set of functional and non-functional requirements to a cloud service provider (datacentre) through a broker's web interface. Then the broker finds the service which best fits the user's request, and ensures that they comply with the user Service Level Agreement (SLA). A large amount of research has taken place in the areas of cloud service discovery and composition, together with tools and techniques which the consumer relies upon to discover and use the services, such as (Nair and Porwal 2010), (L. Zhang, Fowley, and Pahl 2014), (K C Gouda, Radhika T V 2013), (Dinesh, Poornima, and Kiruthika 2012). However, there are still the following limitations in that research area which require immediate attention:

a) They presume that all requested "composed" services reside within a single cloud. Hence, the provider is responsible to integrate these services together to build up the requested one.

b) They do not consider how many services are involved in the composition, which directly impacts on energy consumption and amount of data exchange among services and providers.

QoS metrics are a concern for service providers and brokers, for example service security (Chang, Kuo, and Ramachandran 2016), availability, response time, as these tend to be the aspects of the service which attract clients. The cost of communication, and the sending and receiving of data between composite web services from different cloud providers is expensive and consumes a lot of time and energy. Therefore, this remains a main challenge that this chapter tries to address.

6.2. Model design

To frame the problem and solution, the main parties in the multi-cloud environment must be identified, which are: users, a broker, and service providers. The following sections show how the interrelationship among those parties is formalised, and demonstrates how this new Energy Efficient Cloud Computing service composition algorithm (E2C2) works when identifying and selecting the most energy efficient service composition plan.

6.2.1. User-broker model

The user-broker model is the service request model, which a user sends to a broker to determine the required service. A web service s is syntactically described by two sets of parameters as shown in Fig.18 : $s^i = \{I1, I2, \dots\}$ as input obtained from the service request, and $s^o = \{O1, O2, \dots\}$ as output for the service response. Therefore, the user needs to submit

a service request in a 2-tuple format $\langle I, G \rangle$ where, I is the initial interface indicating the request such that $I \supseteq s^i$; and G is a goal interface, which indicates the ultimate response the user wants to get, such that $G \subseteq s^o$. This makes it an easy route by which to find a single web service matching the user's needs, if and only if $I = s^i$ and $G = s^o$. But should there be a user request where a single web service is unable to meet the request, there then needs to be a composition plan implemented. Thus, in this section the focus is on creating a composition plan for multiple web services in a multi-cloud providers' environment. This algorithm should check cloud providers in descending order based on the number of predefined composition plans.

Once the user request is received, the broker defines the service composition using a four-tuple model $\langle I, G, S, \pi_B \rangle$ where I and G already user defined as previously mentioned, S is a set of candidate web services that are identified by the broker which match with the outcome based on G; and π_B is a composition plan that is a sequence of ordered web services such that $\pi_B \subseteq S$. By applying each service in π_B, the resulting interface is a superset of G.

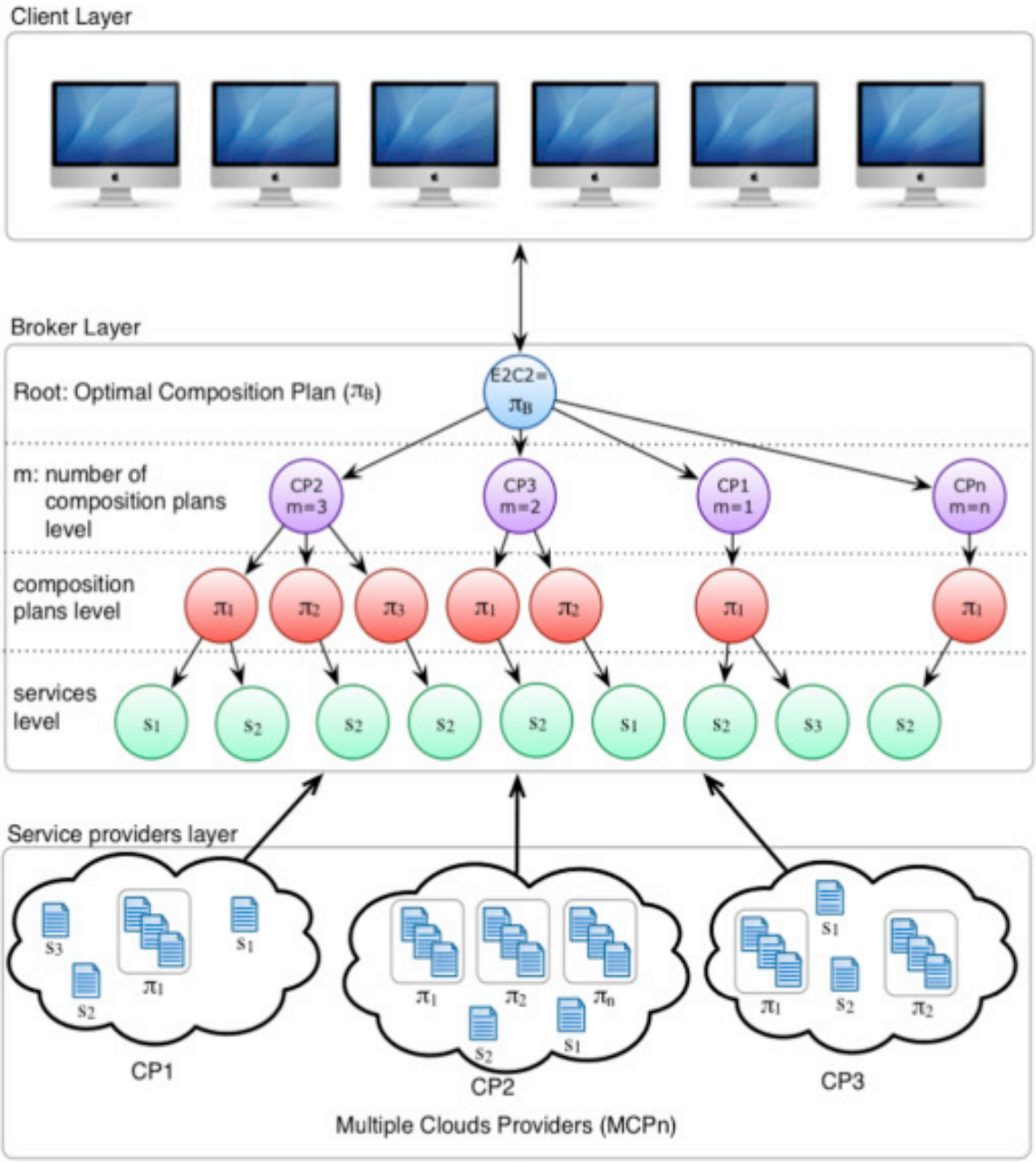

Figure 6.18 A conceptual representation of the proposed approach.

6.2.2. Datacentre-broker model

Each of the requested services could be originated from different commercial cloud providers in a multi-cloud environment. These services can be combined and used together via mutual communication protocols to satisfy a complex service request. Therefore, the Multiple Cloud service Providers (MCP) is a set of cloud providers, such that:

$$MCP = \{CP_{i.\,m}, CP_{i\,+\,1.\,m}, \ldots, CP_{n.\,m}\}.$$

Each CP in MCP is identified by 2 numbers i and m such that $(1 \leq i \leq n)$ represents a CP unique identification number; and m is the number of pre-defined/developed composition plans provided by each specific CP. For purposes of illustration, the $CP3.6$ relates to the Cloud Provider 3 that provides 6 composition plans, in addition to the atomic services. Additionally, $\pi j(CPn.m)$ denotes to the pre-defined composition plan (j) available at $(CPn.m)$.

Because service computation requires energy, this is included in the algorithm, it is thought crucial that service providers provide the broker with each service's energy consumption variable, which allows the broker to decide upon which is the most energy efficient. Therefore, a service s is described by its provider in a 3-tuple format $\langle s^i, s^o, s^{ec} \rangle$, where s^{ec} is the energy required for the service computation at the hosting datacentre. The proposed algorithm makes the following assumptions:

1 The broker creates a composition plan, denoted as (π_B) which includes services from either the same or a different provider.

2 Each of the service providers will send the number of its pre-defined composition plans (m), the actual composition plans $\pi(CP)$, and a list of atomic services in the form of $\langle s^i, s^o, s^{ec} \rangle$.

3 Cloud providers are listed in the proposed algorithm in *descending* order based on (m). This means the cloud provider, which has the largest number of composition plans will be first in the list. This procedure helps to create a final composition plan, which contains the lowest possible number of service providers.

4 The broker first begins to examine the pre-defined composition plans of the provider that hosts the largest number of composition plans to attempt to find one. If no match

is found, the broker will continue the process until a match with the user request is found, as shown in (Algorithm 6.1).

5 If there are no matches to the user request, then individual services, which may be a subset of a pre-defined composition plan, will then be checked, as shown in (Algorithm 6.2).

Figure 6.18 shows how the proposed algorithm works via a high-level conceptual representation, where the root of Broker Layer is the final web service composition plan (π_B). If $\pi_B = \pi_j(CPn.m)$, then π_B is the optimal composition plan that contains the same provider's services, this is also the most energy efficient one amongst all of the composition plans available. Alternatively, the broker creates a composition plan as per the following section Table 6.18.

6.3. Implementation

6.3.1. Optimal service composition plan

As in the above mentioned, S is a collection of candidate web services, which will be firstly identified and titled by the broker to meet the needs of the user. Thus, S can be viewed as $\mathrm{S} = \{ \langle \mathrm{s_i}, \mathrm{CP_p} \rangle , \langle \mathrm{s_j}, \mathrm{CP_q} \rangle , \dots , \langle \mathrm{s_k}, \mathrm{CP_r} \rangle \}$ in a way that applying the web services results in a sequence in an interface R, where $G \subseteq R$. The wanted output o of each service s in S will both result in $(s_1^o \cup s_2^o \cup \dots \cup s_k^o)$. By accounting for the minimum energy efficiency condition of the proposed algorithm, the broker optimal composition will be a sequence of:

$\langle \mathrm{s}', \mathrm{CP}', \mathrm{EC}(\mathrm{s}') \rangle$, such that:

$$\pi'_B = \begin{cases} \min \{\langle s'_i,\ CP'_p, EC(s'_i)\rangle, \langle s'_j,\ CP'_q, EC(s'_j)\rangle, \dots, \langle s'_k,\ CP'_r, EC(s'_k)\rangle\} \\ \subseteq S, \quad subject\ to\ (1)\ below: \\ \pi_j(CP_{n.m}) \end{cases}$$

$$minCons \left| \sum_{i=1}^{k} \{EC(s_i')\} \right| s^i \in \pi_B' \ | \tag{6.1}$$

Therefore, the optimal composition plan π_B' needs to be either a set of atomic services from a different or the same cloud provider that guarantees the minimum number of services possible involved with the minimum energy required to compute each service selected, or a pre-defined composition plan by one of the subscribed cloud providers. In regard to the example in Section 6.1, the same scenario which has been used in Section 5.2.2 of Chapter five is used here to differentiate the approaches explicitly. Assume there is a multi-cloud environment with four cloud providers $\{CP_1, CP_2, CP_3, CP_4\}$ for a broker to deal with. Shown below in Table 6.19, every one of the providers offers a set of atomic services, these are subsets of services $\{a, b, c, d, e\}$, and a set of pre-defined composition plans $\pi(CP)$.

Table 6.13 Multiple-cloud providers and services.

Cloud providers	$CP_{4.4}$	$CP_{1.3}$	$CP_{2.1}$	$CP_{3.1}$
Atomic services	a, b, c, e	a, b, c	c, d, e	c, d
EC (kW)	0.52, 0.8, 0.721, 0.56	0.65, 0.5, 1.2	0.72, 0.32	1.2, 0.45
$\boldsymbol{\pi(CP)}$	{a,e}, {b,c,e}, {c,e}, {b,e}	{a,b}, {a,c}, {b,c}	{d,e}	{c,d}

When the user request ($USR = \langle I, G \rangle$) is received, the broker begins to examine the four subscribed service providers' composition plans. Beginning with the first one, which has the greatest number of composition plans, shown above in Table 6.19 as $CP_{4.4}$. For example, if a user requests services b, c, e, the providers are then checked in the order $CP_{4.4}, CP_{1.3}, CP_{2.1}, CP_{3.1}$.

Therefore, as $CP_{4.4}$ has previously laid out the definition of the requested composition plan, this then satisfies the fewest providers' condition to have involvement in the composition. The broker continues to check the other cloud providers' compositions to discover if there is

another predefined composition plan, which has lower energy consumption and can also satisfy the USR. The suggested E2C2 algorithm brings structure to the multi-cloud environment, the services and any pre-defined composition plans which are available as a four-level tree format, as shown in Figure 6.18, so that:

a) Level 1: The Root, this is the preferred service composition plan by E2C2.
b) Level 2: This re-orders the CP in a descending order based on the number of the pre-defined composition plans by each subscribed CP.
c) Level 3: Lists actual services composition plan(s) by each subscribed provider, together with total energy consumption by each composition plan.
d) Level 4: This is atomic web service in each of the composition plans (e.g., a, b, etc.)

In this instance, the E2C2 algorithm starts ($USR = \langle I, G \rangle$) in line 2 of Algorithm 6.1. From there, the algorithm then selects (in line 3) and checks (in line 6–24) the first CP, which contains the largest number of (m) pre-defined composition plans. This particular method of sorting can help to rapidly find the cloud provider which has the larger number of composition plans, this could be a composition plan which satisfies the request received (as shown in line 14–22). But, it is worth noting that clouds are organised not necessarily in an order such as this in real-life scenarios. When all clouds are arranged and listed, then there is an examination of all composition plans. Should a composition plan be found, the energy that the plan needs is stored in the ($minCons$) buffer. Inside the $minCons$ the values stored are used whenever there is a matching composition plan in order to make a comparison between consumption and the one saved in the $minCons$. On the other hand, if a composition plan cannot be identified from the initial provider, then in turn the next provider is checked (as shown in line 35), this continues until the correct cloud provider is reached.

There is also the chance that the predefined composition plan, which has been identified becomes dynamic at runtime (for example it is a failed or unavailable service(s) needs hot-plugging); in a case such as this Algorithm 6.1 restarts efforts to discover an alternative predefined composition (line 35), or else Algorithm 6.2 begins at line 40 to build an atomic service based dynamic composite plan.

Step 1:

Algorithm 6.1 *discovering a predefined optimal composition plan π'_B from a single provider.*

Algorithm 6.1: Finding a predefined optimal composition plan $\boldsymbol{\pi'_B}$ from a single provider	
Input	User service request (USR), multiple cloud providers (MCP), largest number of composition plan m.
Output	Optimal composition plan π'_B
Assumption	Cloud providers are stored in decreasing order based on the number of composition plans
1	Beginalgorithmic [1] USR ← Ø; π'_B NULL; minCons NULL; m largest number of composition plan; ▷Initialise
2	Get USR $\langle I, G \rangle$
3	Select (CP_m) ▷ CP that contains the largest number of composition plans m
4	$i \leftarrow m$
5	if (i is True) then
6	$foreach\ \ j \leftarrow 1$ to $j \leq i\ step\ 1$ do
7	if $(\pi_j(CP_i) \cap USR == \emptyset\ then$
8	if $(j = i)$ then
9	go to 35
10	else
11	go to 6
12	end
13	else
14	if $(j = 1)$ then
15	$minCons \leftarrow EC(\pi_j(CP_i))$
16	else
17	if $(EC(\pi_j(CP_i)) < minCons)$ then
18	$minCons \leftarrow EC(\pi_j(CP_i))$
19	else
20	go to 6
21	end
22	end
23	end
24	end
25	return $minCons$
26	if $(i = m)$ then
27	$\pi'_B \leftarrow minCons$
28	else
29	if ($minCons < \pi'_B$) then
30	$\pi'_B \leftarrow minCons$
31	end
32	end
33	return π'_B ▷ Optimal composition plan
34	end
35	$i = i - 1$
36	if $(i \geq 1)$ then
37	select (CP_i)
38	go to 5 ▷ So that other CPs will be checked in a decreasing order
39	else
40	invoke algorithm 6.2
41	end

Otherwise, the algorithm can also cope with a situation when a single pre-defined composition plan which satisfies the user's need is not available, Algorithm 6.2 begins to check available composition plans' services to see if any match a subset of services requested by the user, defined in USR and shown in line 6 of Algorithm 6.2. Additionally, providers' atomic services are checked and any that match the user's request are combined, as shown in line 16. The number of providers collaborating in the final composition (line 15) is also calculated, and then total energy consumption of the final composition is calculated, displayed in line 14.

Step 2:

algorithm 6.2 *The creation of a dynamic optimal composition plan π'_B from multiple providers.*

```
1  Get (USR, ⟨I, G⟩
2  Select (CP_m )
3  i ← m
4  if (i is True) then
5    foreach j ← 1 to j ≤ i  step 1 do
6       if (π_j(CP_i) ∩ USR) ⊂ USR then
7            if (i = m) then
8                 eneCons ← EC(π_j(CP_i))
9            else
10                if (EC (π_j(CP_i)) < eneCons then
11                   eneCons ← EC(π_j(CP_i))
12                end
13           end
14           minCons = minCons + eneCons
15           proList = proList + CP_i
16           serList = serList ∪ ((π_j(CP_i) ∩ USR))
17      else
18           if (π_j(CP_i) ∩ USR) == ∅ then
19              if (j = i) then
20                   go to 27
21              else
22                   go to 5
23              end
24          end
25      end
26  end
27 else
28  i = i − 1
29  if (i ≥ 1) then
30       Select ⟨CP_i⟩
31       go to 4
32  else
33       exit
34  end
35 end
```

Functionality of the resources is described and published by the service registry; they are presented to possible customers bundled as web services. Our proposed algorithm can be used by a service broker to create an optimal composition plan by searching for and locating the service registry for the most energy efficient services. Matching users' functional requirements is not the only purpose of service discovery, energy efficiency is also considered to help in achieving the green cloud environment. A cloud carrier (Mell and Grance 2011) provides connectivity between cloud entities by ensuring seamless provisioning. In this instance, Internet is one of the better methods by which to access and use cloud services. The purpose of cloud auditors (Mell and Grance 2011) can be included into the suggested framework to perform a design time verification of the energy efficient composition which is generated and is then evaluated for how adaptable it is toward runtime changes. At this point, the cloud auditor's role is considered beyond this paper's scope.

6.4 Evaluation

To evaluate E2C2 performance and efficiency, it is crucial to build a comparison of the results against well-established benchmark algorithms so that the potential for reducing energy consumption can be measured. This comparison was carried out using some of the existing algorithms used in evaluating combinations of cloud services: (All Clouds, Base Cloud, Smart Cloud (Zou et al. 2010) and COM2 (Kurdi et al. 2015).

6.4.1. Experimental settings

As mentioned above, for our comparative evaluation purposes we adopted four different algorithms for selecting the cloud services combination: (All Clouds, Base Cloud, Smart Cloud (Zou et al. 2010) and COM2 (Kurdi et al. 2015). The first algorithm, All Clouds, suggests that all clouds should be inputs for the composition and all available solutions should be determined. The algorithm assists in the location of a service composition

sequence, which has the smallest execution time, without minimising the number of clouds in the final composition. All possibilities of cloud combination are repeatedly enumerated by the Base Cloud algorithm, in increasing order or until an ideal solution is found.

The process starts by the analysis of all singleton sets of clouds, it ceases if the combination needed is discovered utilising a single cloud, alternatively it will extend the search to cloud sets of size two, then three, up until the point that the required combination is discovered. It produces the best composition solution from the fewest clouds. The Smart Cloud algorithm was produced to find a near optimal composition plan based on an approximation algorithm. A multiple cloud environment is considered a tree, which then searches the tree to identify a minimum demand set.

A near-optimal solution is located by the Smart Cloud at a lower cost while using a reduced cloud set. Kurdi et al., (2015) suggested that a unique combinatorial optimisation algorithm could consider multiple clouds and perform a service composition with the fewest clouds and with the shortest execution time, this has the benefit of a reduction in communication costs.

In the E2C2 simulation a comparative evaluation was performed using the aforementioned algorithms and identical simulation parameters of the four algorithms. The data from the experiment was inspired by the default web service test-set provided in the OWL-S XPlan package (Klusch and Gerber 2006)(Karunamurthy, Khendek, and Glitho 2012), along with high-level semantic and syntactic declarative descriptions of service properties (Nacer and Aissani 2014) in terms of $\langle I, G \rangle$ to find out if the desired services are compostable. The experiments used an Apple iMac (Retina 5 K display, 3.2 GHz Intel Core i5, and 8 GB 1867 MHz DDR3). The prototype development platform and the simulation running environment was NetBeans as the Integrated Development Environment (IDE), and Java EE 8 was used as the programming language to implement the proposed algorithm.

There are four cloud providers in our simulation $\{CP_1, CP_2, CP_3, CP_4\}$. Each provider gives a set of pre-defined composition plans, each of which are subsets of $\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$, and are based on the Multi-cloud providers (MCP) environment as shown in Table 6.21. The same scenario used in Section 5.4.1. of Chapter 5 is used here to demonstrate the difference of listing the cloud providers discerningly based on number of composition plans.

Table 6.14 Cloud providers composition set per multiple-cloud providers environment.

MCPs	CP_1	CP_2	CP_3	CP_4
MCP_1	π_1, π_2, π_3	π_4, π_5	π_3, π_4	$\pi_1, \pi_2, \pi_3, \pi_5$
MCP_2	π_1, π_2	π_3	π_2, π_5	π_1, π_4, π_5
MCP_3	π_1, π_3, π_5	π_5	π_1, π_2	π_3, π_4
MCP_4	π_2, π_3, π_5	π_3, π_4	π_1, π_2, π_3	π_4, π_5
MCP_5	π_1, π_2	π_2, π_3	π_3	π_1, π_4, π_5

Additionally, {2,3,8,3,3} represents how many web services are involved in each of the composition plans respectively, as shown in Table 6.22.

Table 6.15 Number of services per composition plans

Composition plan	π_1	π_2	π_3	π_4	π_5
Number of services	2	3	8	3	3

The list shows the cloud providers listed in *descending* order, based on the total number of pre-defined composition plans provided. This can assist in getting a final composition plan with the fewest providers and services involved, should there not be a single pre-defined composition plan, which satisfies the request. It should be said that the order of this is different in each MCP environment, for the same provider, dependent upon the quantity of composition plans, demonstrated in Table 6.23.b. As an example, CP4 comes first in MCP1

because there are four composition plans, but it comes last in MCP4 with only 2 composition plans.

Table 6.16 CPs and number of (π) composition plans per MCPs.

(a) Before descending order of CPs				
MCP_S	CP_1	CP_2	CP_3	CP_4
MCP_1	$CP_{1.3}$	$CP_{2.2}$	$CP_{3.2}$	$CP_{4.4}$
MCP_2	$CP_{1.2}$	$CP_{2.1}$	$CP_{3.2}$	$CP_{4.3}$
MCP_3	$CP_{1.3}$	$CP_{2.1}$	$CP_{3.2}$	$CP_{4.2}$
MCP_4	$CP_{1.3}$	$CP_{2.2}$	$CP_{3.3}$	$CP_{4.2}$
MCP_5	$CP_{1.2}$	$CP_{2.2}$	$CP_{3.1}$	$CP_{4.3}$

(b) After descending order based on number of π				
MCP_S	**Sorting order of CPs**			
MCP_1	$CP_{4.4}$	$CP_{1.3}$	$CP_{2.2}$	$CP_{3.2}$
MCP_2	$CP_{4.3}$	$CP_{1.2}$	$CP_{3.2}$	$CP_{2.1}$
MCP_3	$CP_{1.3}$	$CP_{3.2}$	$CP_{4.2}$	$CP_{2.1}$
MCP_4	$CP_{1.3}$	$CP_{3.3}$	$CP_{2.2}$	$CP_{4.2}$
MCP_5	$CP_{4.3}$	$CP_{1.2}$	$CP_{2.2}$	$CP_{3.1}$

6.4.2 Results and analysis

The results gained from simulating the four benchmark algorithms confirm to the previously published results, All Clouds in Table 25.a, the Base Cloud in Table 6.24.b, the Smart Cloud in Table 6.24.c and COM2 in Table 6.24.d. (Zou et al. 2010) (Kurdi et al. 2015). The results are listed in Table 6.24.e, showing E2C2 performance, so that they can be compared with the aforementioned approaches. As in previous studies, the same cloud simulation environment and simulation parameters were used. In the experiments we conducted, the three performance measures considered are:

a) The number of cloud providers which are featured in the final composition |CP|.

b) The number of services checked before the final composition was achieved |S|.

c) The running time, measured in seconds, it took the algorithm to run until an appropriate composition is achieved.

Table 6.17 CPs and number of (π) composition plans per MCPs.

(a) All Clouds Algorithm			
Performance	**CP involved**	**|C P|**	**|S|**
MCP1	CP1 CP2 CP4	3	46
MCP2	CP1 CP2 CP3 CP4	4	27
MCP3	CP1 CP3 CP4	3	32
MCP4	CP1 CP2 CP3 CP4	4	44
MCP5	CP1 CP2 CP3 CP4	4	32
Total		18	181

(b) Based Cloud Algorithm			
Performance	**CP involved**	**|C P|**	**|S|**
MCP1	CP1 CP2	2	65
MCP2	CP1 CP2 CP4	3	148
MCP3	CP3 CP4	2	128
MCP4	CP2 CP3	2	68
MCP5	CP2 CP4	2	112
Total		11	521

(c) Smart Cloud Algorithm			
Performance	**CP involved**	**|C P|**	**|S|**
MCP1	CP1 CP3	2	70
MCP2	CP1 CP2 CP4	3	48
MCP3	CP3 CP4	2	48
MCP4	CP2 CP3	2	140
MCP5	CP1 CP2 CP4	3	56
Total		12	362

(d) COM2 Algorithm			
Performance	**CP involved**	**|C P|**	**|S|**
MCP1	CP4 CP2	2	35
MCP2	CP4 CP2 CP3	3	45
MCP3	CP1 CP4 CP3	3	50
MCP4	CP1 CP3 CP2	3	49
MCP5	CP2 CP4	2	30
Total		13	209

(e) E2C2 Algorithm			
Performance	**CP involved**	**|CP|**	**|S|**
MCP1	CP4 CP2	2	35
MCP2	CP4 CP1 CP2	3	26
MCP3	CP1 CP3 CP4	3	29
MCP4	CP1 CP3 CP2	3	38
MCP5	CP4 CP2	2	24
Total		12	152

In Table 6.24 the results show that the E2C2 algorithm was successful in surpassing all of the other algorithms in keeping a lower number of composite clouds and examined services. This can also be directly related to the execution time, which is much less than the execution time of the best of the four algorithms, which is COM2. The number of atomic services which were examined |S| was not more than 38. The top result, relating to the difference in the number of examined services in E2C2 and the best algorithm in each of the five environments, have been reached in MCP4 and MCP5. There were 6 fewer atomic services, which were examined by E2C2 in comparison to All Clouds in MCP4, and the same (6 fewer atomic services) between E2C2 and COM2 in MCP5. Figure 6.19 details a comparison of the % reduction in the number of examined atomic services (relative to the baseline case for each MCP) by E2C2, All Clouds, Base Cloud, Smart Cloud, and COM2. Each MCP's baseline relates to the case of the maximum number of atomic services, which were examined for that specific MCP. When calculating the percentage reduction, this can be used as a reference point. Figure 6.20 compares the average number of atomic services examined in E2C2 across all MCPs, which is 30 services, in comparison to the average number of examined services of all of the other algorithms.

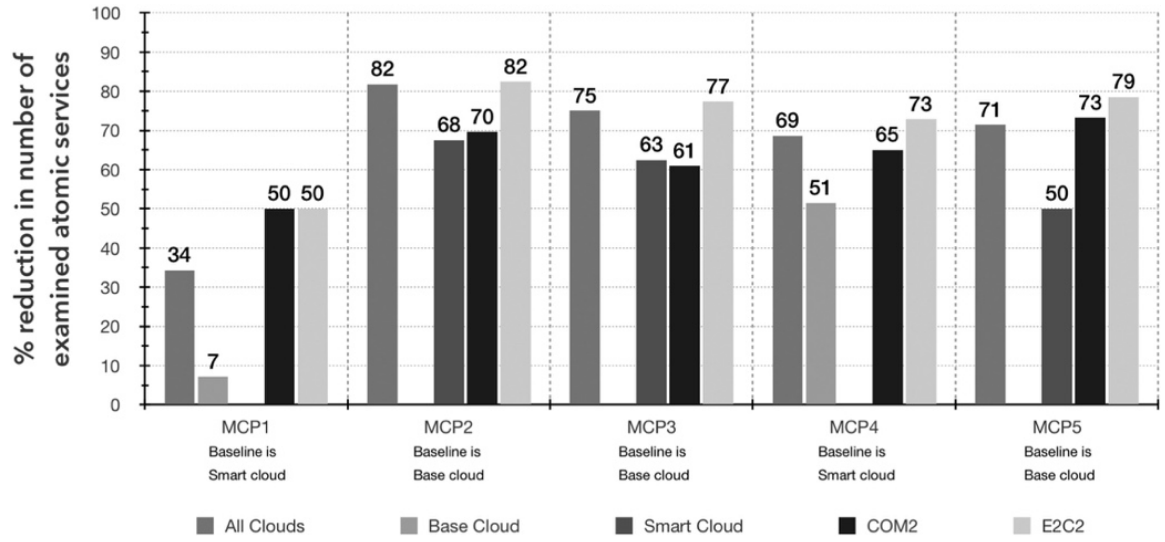

Figure 6.19 % reduction in the number of examined atomic services.

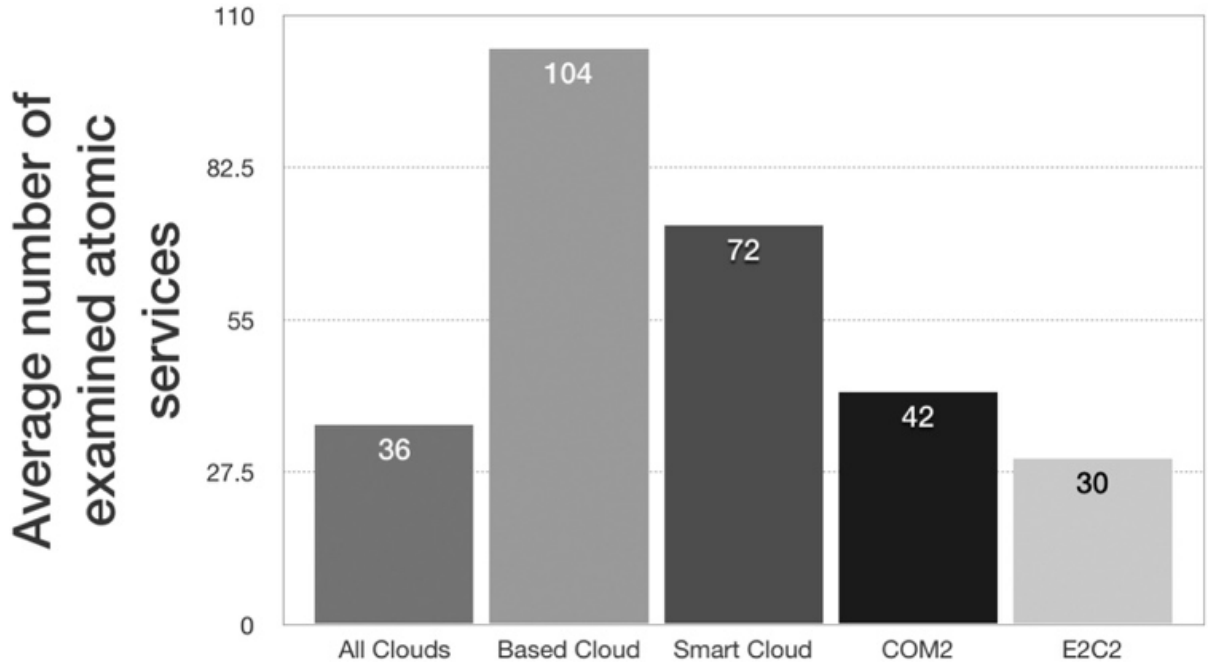

Figure 6.20 The average number of examined services.

In our E2C2 algorithm the number of combined clouds occasionally reached as low as two clouds but was never more than three in the worst case, as shown in Figure 6.21. Even though Table 6.24.b demonstrates that the Base Cloud algorithm is the best in terms of the

number of combined clouds, it is weak because of the high costs in execution time and number of examined services, this can be over 3 times the number of services in the projected E2C2 algorithm. We get these results from a pre-sorted list of cloud providers which are in decreasing order based on the number of composition plans. Compared to COM2 there was a substantial performance improvement in E2C2 execution time, as shown in Figure 6.22, of the four algorithms examined in each MCP, this is the best. Figure 6.22 displays results that show the execution time of E2C2 did not exceed 173 seconds in the best scenario (in MCP5), and did not exceed 298 seconds in the worst scenario (in MCP4).

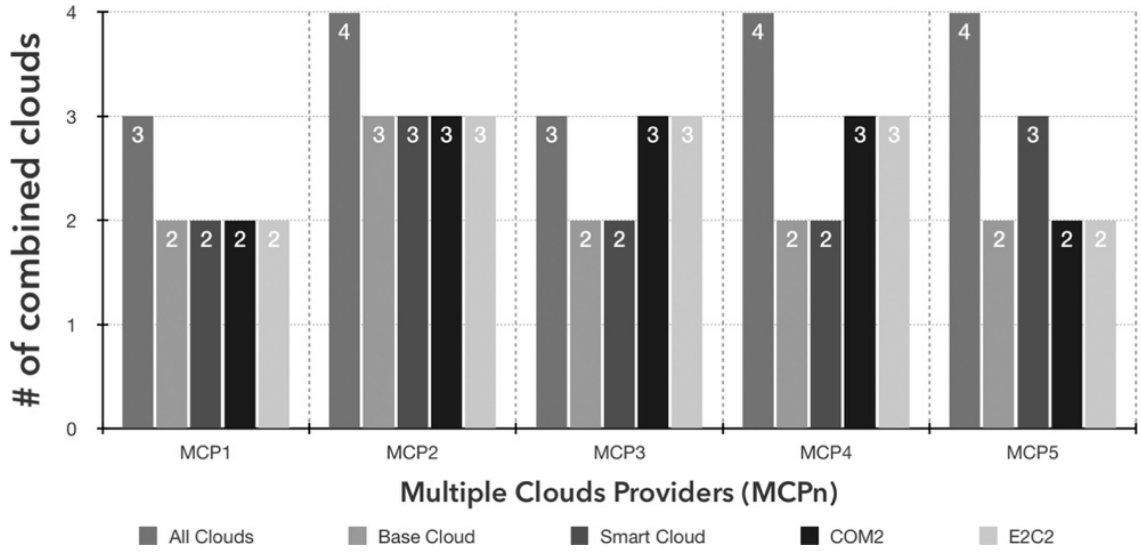

Figure 6.21 Number of combined clouds in multi-cloud based service composition plan π.

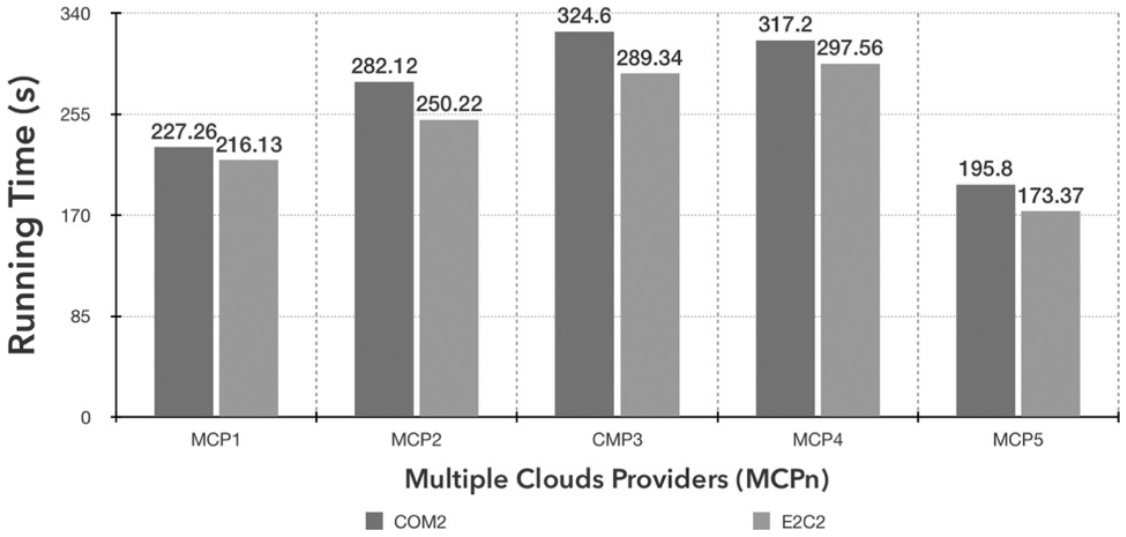

Figure 6.22 Running Time to achieve service composition plan π.

6.5. Summary

This chapter introduced a unique multi-cloud service composition algorithm named E2C2. This algorithm highlighted energy awareness during searches for optimum composition plans, which meet the specified requirements of the user. It conducted a search for and then integrated the fewest services which satisfy user requests in an energy efficient way. This algorithm was evaluated based on the systematic performance comparison with existing alternative algorithmic solutions, such as All Cloud, Base Cloud, Smart Cloud and COM2, using a benchmark example. The chapter concludes with the simulation results that demonstrated a favourable performance of E2C2 algorithm in terms of achieving the least number of services searched to arrive at an optimum and energy efficient composition.

Conclusion and Future Work

6.1. Conclusion

The growing demand of cloud services, and the increase in service providers and data centres to offer and host cloud application across every geographical region have led to significant increases in network traffic and the associated energy consumed by the huge infrastructure (e.g., extra servers and switches) required to respond quickly and effectively to user requests. Moreover, transporting data between data centres and users can consume even larger amounts of energy than just processing and storing the data on the datacentre itself. Therefore, energy efficiency has become a high priority aim in cloud network environment. This power consumption is particularly significant when transferring data into a datacentre located relatively far from the user geographical location. In this research of this we presented and demonstrated numbers of algorithms that help to achieve the aim of this thesis to select the most energy efficient route to the cloud data centre alongside finding and integrating the lowest possible number of the most energy efficient services from the least possible number of cloud providers which can help in achieving a full green cloud computing environment, which eventually contributes to the energy saving goals by 2050.

In this research, we propose and evaluate a multi-cloud broker to act as an intermediary bridge for directing users' requests to the green data centres using the most energy efficient route. In addition, the broker manages the cloud services to achieve the full green cloud computing network ambition. The GreeDi algorithm dealt with the energy efficiency of cloud routing rather than data centres' energy consumption, and proposes and evaluates a new energy efficient routing framework. It was evaluated on a physical Italian ISP topology that has three different routes to a green cloud data centre. From the example results shown in this research, the shortest path approach is different from the energy efficient one, and thus, the energy efficient path is used to conform to the environmental objectives.

A novel multi-cloud service computing approach is presented, focusing on the selection of energy efficient services and service composition plans that meet user requirements. In particular, our Bin-packing based service composition algorithm determines the minimum number of composite services based on an effective combination of cloud services providers that satisfy the user needs. Our approach therefore addresses the increasing need for optimising energy consumption associated with the rise in complex (or real-world) cloud based service and user request scenarios, which are characterised by a large number of cloud providers and services. The proposed approach was evaluated against five established service computing algorithms for multiple cloud environments and the simulation results demonstrated that our algorithm produces significant relative performance improvements in terms of both running time and energy consumption.

E2C2 is a unique multi-cloud service composition algorithm, its development aimed to highlight energy awareness during searches for optimum composition plans which meet the specified requirements of the user. This algorithm conducts a search for and then integrates the fewest services which satisfy user requests in an energy efficient way. The algorithm was evaluated based on the systematic performance comparison with existing alternative algorithmic solutions, namely All Cloud, Base Cloud, Smart Cloud and COM2, using a benchmark example.

6.2. Contributions to knowledge

The main contribution of this work is the development of a multi-cloud broker and associated algorithms and models to achieve the most energy-efficient cloud computing environment. To this end, this work makes a number of novel contributions, all of which have been, or are being, submitted to relevant research publications (Appendix I). Contributions of this work are summarised as follows:

- Collation of researches relevant to understanding the fundamental requirements to enable the managing of cloud services in a multi-cloud environment and achieve the most energy-efficient cloud computing environment.
- Design, and development of a high-end routing algorithm entitled Green Director (GreeDi), which acts as an intermediary bridge for directing the users' requests to the green data centres based primarily on using the most energy efficient route to achieve the full green cloud computing network ambition while making sure the users' requirements, e.g. response time, are met.
- Defining and formalising the user-to-broker and data centre-to-broker models so that, the interconnection between the cloud user and a data centre, will be formalised by using a situation calculus model to define the logical state of the network. Once the interconnection is established and formalised, then the energy required for the transportation will be calculated.
- Design, and development of a bin-packing based energy-efficient service selection algorithm that operates by way of Integer Linear Programming (ILP). The aim of this model is to find and integrate, from the largest possible number of service providers, the smallest possible number of services offering the highest level of energy efficiency. To achieve this approach, the following algorithms should be designed:
    i. An algorithm to rank cloud service providers by total energy consumption.
    ii. An algorithm to enable the broker to search for the best possible match by seeking a combination of individual atomic service and lowest energy consumption.

iii. An algorithm to allow the broker to search predefined composition plans and their energy consumption in order to find the best possible match for user requirements.

iv. An algorithm to allow the broker to build a new optimal composition plan by selecting the most energy efficient services from the smallest possible number of providers.

- Design, and development of a unique energy-aware multi-Cloud service composition algorithm called (E2C2) which develops energy efficient composition plans through the integration of the fewest services globally from service providers. Therefore, this work has the objective of addressing an emerging and key issue that adopts an energy efficient approach for cloud-based services and applications.
- Evaluation of the above algorithms supported by experiments and case studies, and comparison against renowned competitors. To achieve this approach, the following algorithms should be designed

    i. An algorithm to discover a predefined optimal composition plan from a single provider.

    ii. An algorithm to create a dynamic optimal composition plan from multiple providers.

6.3. Future work

The future work will be to extend this work and develop potential energy efficiency gains in the cloud computing and will be more focusing on following:

- Putting together these algorithms in one model so that we can test and evaluate the whole system.

- Future extensions to the GreeDi algorithm include analysing and taking into account the time required for transportation, and energy and time required for computation, between the data centres and users and among the data centres themselves, in order to establish and evaluated how the proposed algorithm performs in terms of computation consumption.
- Developing solutions in other application domains (e.g., the applicability/viability of GreeDi used in underwater routing, and comparison against Fast Multi-path Routing Protocol for wireless sensor networks (FMRP)) using our GreeDi to examine to what extent GreeDi is applied for.
- Evaluating our Bin-packing based service composition algorithm using other complex service selection scenarios and assessing its applicability and performance in key application domain such as smart cities and mobile commerce.
- Also, we consider evaluating the other case of Bin-packing where it orders the cloud providers in a descending way based on the total energy consumption of their corresponding services.
- Exploration of modifications to the E2C2 algorithm, by testing the algorithmic performance using more elaborate topologies.
- Other paths for future research include examining the potential benefit of heuristic optimization search techniques for selecting service compositions and evaluating the potential energy efficiency gains in various application domains such as mobile commerce, smart government and disaster recovery scenarios.

7. References

Aldawsari, Bandar, Thar Baker, and David England. 2015. “Towards a Holistic Multi-Cloud Brokerage System: Taxonomy, Survey, and Future Directions.” In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 1467–72. IEEE. doi:10.1109/CIT/IUCC/DASC/PICOM.2015.219.

Ali Alakeel. 2012. “A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems.” *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 6 (1). http://waset.org/publications/15228/a-fuzzy-dynamic-load-balancing-algorithm-for-homogenous-distributed-systems.

AlShawi, Imad S., Lianshan Yan, Wei Pan, and Bin Luo. 2012. “Lifetime Enhancement in Wireless Sensor Networks Using Fuzzy Approach and A-Star Algorithm.” *IEEE Sensors Journal* 12 (10): 3010–18. doi:10.1109/JSEN.2012.2207950.

Amazon. 2017. “Elastic Compute Cloud (EC2) – Cloud Server ; Hosting – AWS by Amazon.” Accessed March 15. https://aws.amazon.com/ec2/.

Baer, Paul. 2008. “Exploring the 2020 Global Emissions Mitigation Gap Analysis for the Global Climate Network.” http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.505.7686&rep=rep1&type=pdf.

Baker, T., B. Al-Dawsari, H. Tawfik, D. Reid, and Y. Ngoko. 2015. “GreeDi: An Energy Efficient Routing Algorithm for Big Data on Cloud.” *Ad Hoc Networks*, June. doi:10.1016/j.adhoc.2015.06.008.

Baker, Thar, Yanik Ngoko, Rafael Tolosana-Calasanz, Omer F. Rana, and Martin Randles. 2013. “Energy Efficient Cloud Computing Environment via Autonomic Meta-Director Framework.” In *6th International Conference on Developments in eSystems Engineering*, 198–203. IEEE. doi:10.1109/DeSE.2013.43.

Baliga, J, R W A Ayre, K Hinton, and R S Tucker. 2011. “Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport.” *Proceedings of the IEEE* 99 (1): 149–67. doi:10.1109/JPROC.2010.2060451.

Banimelhem, Omar, and Samer Khasawneh. 2012. “GMCAR: Grid-Based Multipath with Congestion Avoidance Routing Protocol in Wireless Sensor Networks.” *Ad Hoc Networks* 10 (7): 1346–61. doi:10.1016/j.adhoc.2012.03.015.

Bartalos, Peter, and M. Brian Blake. 2012. “Engineering Energy-Aware Web Services toward Dynamically-Green Computing.” In , 87–96. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-31875-7_10.

Beloglazov, Anton, Jemal Abawajy, and Rajkumar Buyya. 2012. “Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing.” *Future Generation Computer Systems* 28: 755–68. doi:10.1016/j.future.2011.04.017.

Beloglazov, Anton, and Rajkumar Buyya. 2010. “Energy Efficient Resource Management in Virtualized Cloud Data Centers.” In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 826–31. IEEE. doi:10.1109/CCGRID.2010.46.

Beloglazov, Anton, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. 2011. “A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems.” *ADVANCES IN COMPUTERS* 82. doi:10.1016/B978-0-12-385512-1.00003-7.

Bhatia, Randeep, Fang Hao, Murali Kodialam, and T.V. Lakshman. 2015. “Optimized Network Traffic Engineering Using Segment Routing.” In *2015 IEEE Conference on Computer Communications (INFOCOM)*, 657–65. IEEE. doi:10.1109/INFOCOM.2015.7218434.

Blackburn, Mark, Dan Azevedo, Andy Hawkins, Zeydy Ortiz, Roger Tipley, and Dr. Sven Van Den Berghe. 2010. “The Green Grid Data Center Compute Efficiency Metric: DCcE | The Green Grid.” https://www.thegreengrid.org/en/resources/library-and-tools/240-The-Green-Grid-Data-Center-Compute-Efficiency-Metric%3A-DCcE.

Brown, Richard, Eric Masanet, Bruce Nordman, Bill Tschudi, Arman Shehabi, John Stanley, Jonathan Koomey, et al. 2007. “Report to Congress on Server and Data Center Energy Efficiency.” https://eetd.lbl.gov/sites/all/files/pdf_4.pdf.

Carpa, Radu, Olivier Gluck, and Laurent Lefevre. 2014. “Segment Routing Based Traffic Engineering for Energy Efficient Backbone Networks.” In *2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 1–6. IEEE. doi:10.1109/ANTS.2014.7057272.

Chang, Victor, Yen-Hung Kuo, and Muthu Ramachandran. 2016. “Cloud Computing Adoption Framework: A Security Framework for Business Clouds.” *Future Generation Computer Systems* 57: 24–41. doi:10.1016/j.future.2015.09.031.

Chang Ge, Zhili Sun, and Ning Wang. 2013. “A Survey of Power-Saving Techniques on Data Centers and Content Delivery Networks.” *IEEE Communications Surveys & Tutorials* 15 (3): 1334–54.

doi:10.1109/SURV.2012.102512.00019.
Chen, Gong, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. 2008. “Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services.” *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 337–50. http://dl.acm.org/citation.cfm?id=1387613.
Cheng, Jiujun, Cong Liu, MengChu Zhou, Qingtian Zeng, and Antti Yla-Jaaski. 2015. “Automatic Composition of Semantic Web Services Based on Fuzzy Predicate Petri Nets.” *IEEE Transactions on Automation Science and Engineering* 12 (2): 680–89. doi:10.1109/TASE.2013.2293879.
Chimakurthi, Lskrao, and Madhu Kumar S D. 2011. “Power Efficient Resource Allocation for Clouds Using Ant Colony Framework,” February. http://arxiv.org/abs/1102.2608.
Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu. 2011. “Energy-Efficient Virtual Machine Provision Algorithms for Cloud Systems.” In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 81–88. IEEE. doi:10.1109/UCC.2011.21.
Clarence Filsfils, Stefano Previdi, Bruno Decraene. 2017. “Segment Routing Architecture, Internet-Draft.”
Coffman, Edward G., Edward G. Coffman, Jr., János Csirik, and Gerhard J. Woeginger. 1999. “Approximate Solutions to Bin Packing Problems.” *WOE-29, INSTITUT FR MATHEMATIK B, TU GRAZ, STEYRERGASSE 30, A-8010*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.6304.
Cohen, Reuven, Shlomo Havlin, and Daniel ben-Avraham. 2002. “Efficient Immunization Strategies for Computer Networks and Populations,” July. doi:10.1103/PhysRevLett.91.247901.
Davoli, Luca, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. 2015. “Traffic Engineering with Segment Routing: SDN-Based Architectural Design and Open Source Implementation,” June. http://arxiv.org/abs/1506.05941.
Deepak Kapgate et al. 2014. “Efficient Service Broker Algorithm for Data Center Selection in Cloud Computing.” *International Journal of Computer Science and Mobile Computing* 3 (1): 355–65. https://www.academia.edu/5847314/Efficient_Service_Broker_Algorithm_for_Data_Center_Selection_in_Cloud_Computing.
DepartmentforTransport. 2003. “Our Energy Future - Creating a Low Carbon Economy.” http://webarchive.nationalarchives.gov.uk/+/http:/www.berr.gov.uk/files/file10719.pdf.
Dinesh, K., G. Poornima, and K. Kiruthika. 2012. “Efficient Resources Allocation for Different Jobs in Cloud.” *International Journal of Computer Applications* 56 (10): 30–35. doi:10.5120/8928-3005.
Do, T. V. 2011. “Comparison of Allocation Schemes for Virtual Machines in Energy-Aware Server Farms.” *The Computer Journal* 54 (11). Oxford University Press: 1790–97. doi:10.1093/comjnl/bxr007.
Do, Tien Van, and Udo R. Krieger. 2009. “A Performance Model for Maintenance Tasks in an Environment of Virtualized Servers.” In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, 931–42. Springer-Verlag. doi:10.1007/978-3-642-01399-7_73.
Do, Tien Van, and Csaba Rotter. 2012. “Comparison of Scheduling Schemes for on-Demand IaaS Requests.” *Journal of Systems and Software* 85 (6): 1400–1408. doi:10.1016/j.jss.2012.01.019.
Docker. 2017. “Docker - Build, Ship, and Run Any App, Anywhere.” Accessed March 15. https://www.docker.com/.
Ducrocq, Tony, Michaël Hauspie, Nathalie Mitton, and Nathalie Mitton. 2013. “Balancing Energy Consumption in Clustered Wireless Sensor Networks.” *ISRN Sensor Networks* 2013 (November). Hindawi: 1–14. doi:10.1155/2013/314732.
editor. 2011. “History of Virtualization | Everything VM.” http://www.everythingvm.com/content/history-virtualization.
Elshaafi, Hisain, and Dmitri Botvich. 2012. “Trustworthiness Inference of Multi-Tenant Component Services in Service Compositions.” In , 301–12. Springer, Dordrecht. doi:10.1007/978-94-007-5699-1_31.
“Eucalyptus - Private / Hybrid Cloud Solution.” 2017. Accessed March 16. http://www8.hp.com/us/en/cloud/helion-eucalyptus.html.
Eunjeong Park, and Heonshik Shin. 2008. “Reconfigurable Service Composition and Categorization for Power-Aware Mobile Computing.” *IEEE Transactions on Parallel and Distributed Systems* 19 (11): 1553–64. doi:10.1109/TPDS.2008.107.
Ferreto, Tiago C., Marco A.S. Netto, Rodrigo N. Calheiros, and César A.F. De Rose. 2011. “Server Consolidation with Migration Control for Virtualized Data Centers.” *Future Generation Computer Systems* 27 (8): 1027–34. doi:10.1016/j.future.2011.04.016.
Forrester. 2012. “Cloud Brokers Will Reshape The Cloud- Getting Ready For The Future Cloud Business.” http://liberatedcloud.com/category/cloud-brokers-will-reshape-the-cloud/.
Fowley, Frank, Claus Pahl, and Li Zhang. 2013. “A Comparison Framework and Review of Service Brokerage Solutions for Cloud Architectures.” http://doras.dcu.ie/19640/.
G. Apostolopoulos,D. Williams,S. Kamat, Lucent, R. Guerin, UPenn, A. Orda, Technion, and T. Przygienda.

1999. “QoS Routing Mechanisms and OSPF Extensions.” Clarendon Press. https://tools.ietf.org/html/rfc2676.
Garriga, Martín, Andres Flores, Alejandra Cechich, and Alejandro Zunino. 2015. “Web Services Composition Mechanisms: A Review.” *IETE Technical Review* 32 (5). Taylor & Francis: 376–83. doi:10.1080/02564602.2015.1019942.
Gartner. 2013. “Cloud Services Brokerage (CSB) - Gartner.” http://www.gartner.com/it-glossary/cloud-services-brokerage-csb/.
GoogAppEngine. 2017. “Google App Engine - Build Scalable Web ; Mobile Backends in Any Language | Google Cloud Platform.” Accessed March 15. https://cloud.google.com/appengine/?utm_source=google&utm_medium=cpc&utm_campaign=2017-q1-cloud-emea-gcp-bkws-freetrial&gclid=CjwKEAjwzKPGBRCS55Oe46q9hCkSJAAMvVuMATK2qBH3r7vqRhDk4_iNtxB40dsYxbu0ZDDj48_zDxoCLf_w_wcB.
GoogleDocs. 2017. “Google Docs - Create and Edit Documents Online, for Free.” Accessed March 15. https://www.google.co.uk/docs/about/.
GSuite. 2017. “G Suite - Gmail, Docs, Calendar, & Cloud Storage.” Accessed March 15. https://gsuite.google.com/intl/en_uk/products/.
Guenter, Brian, Navendu Jain, and Charles Williams. 2011. “Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning.” In *2011 Proceedings IEEE INFOCOM*, 1332–40. IEEE. doi:10.1109/INFCOM.2011.5934917.
Guérout, Tom, Thierry Monteil, Georges Da Costa, Rodrigo Neves Calheiros, Rajkumar Buyya, and Mihai Alexandru. 2013. “Energy-Aware Simulation with DVFS.” *Simulation Modelling Practice and Theory* 39 (December): 76–91. doi:10.1016/j.simpat.2013.04.007.
“Haizea - An Open Source VM-Based Lease Manager.” 2017. Accessed March 16. http://haizea.cs.uchicago.edu/.
Han, Yufei, and Fabien Moutarde. 2012. “Statistical Traffic State Analysis in Large-Scale Transportation Networks Using Locality-Preserving Non-Negative Matrix Factorization,” December. http://arxiv.org/abs/1212.5264.
Hang, Chung-Wei, Anup K. Kalia, and Munindar P. Singh. 2012. “Behind the Curtain: Service Selection via Trust in Composite Services.” In *2012 IEEE 19th International Conference on Web Services*, 9–16. IEEE. doi:10.1109/ICWS.2012.96.
Hermenier, Fabien, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. 2009. “Entropy: A Consolidation Manager for Clusters.” In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments - VEE ’09*, 41. New York, New York, USA: ACM Press. doi:10.1145/1508293.1508300.
Hyper-v. 2017. “Hyper-V | Microsoft.” Accessed March 15. https://www.microsoft.com/en-us/cloud-platform/server-virtualization.
“IBM ILOG CPLEX Optimization Studio.” 1AD. https://www.ibm.com/us-en/marketplace/ibm-ilog-cplex.
Issarny, Valérie., Richard E. Schantz, and Anindya Neogi. 2008. *pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems*. *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer. http://dl.acm.org/citation.cfm?id=1496966.
Jay Lyman. 2014. “451 Research-"Is PaaS Becoming Just a Feature of IaaS"?” https://451research.com/report-short?entityId=79800.
Jebalia, Maha, Asma Ben Letaifa, Mohamed Hamdi, and Sami Tabbane. 2013. “A Comparative Study on Game Theoretic Approaches for Resource Allocation in Cloud Computing Architectures.” In *2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 336–41. IEEE. doi:10.1109/WETICE.2013.11.
Jemili, Imen, Ghazi Tekaya, and Abdelfettah Belghith. 2014. “A Fast Multipath Routing Protocol for Wireless Sensor Networks.” In *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, 747–54. IEEE. doi:10.1109/AICCSA.2014.7073275.
Jos G.J. Olivier, Greet Janssens-Maenhout, Jeroen A.H.W. Peters, Julian Wilson. 2011. “Long-Term Trend in Global CO2 Emissions.” http://www.pbl.nl/sites/default/files/cms/publicaties/C02 Mondiaal_ webdef_19sept.pdf.
K C Gouda, Radhika T V, Akshatha M. 2013. “Priority Based Resource Allocation Model for Cloud Computing.” *International Journal of Science, Engineering and Technology Research* 2 (1): 215–19.
Kacimi, Rahim, Riadh Dhaou, and André-Luc Beylot. 2013. “Load Balancing Techniques for Lifetime Maximizing in Wireless Sensor Networks.” *Ad Hoc Networks* 11 (8): 2172–86. doi:10.1016/j.adhoc.2013.04.009.
Kapgate, Deepak Dashrath, and Manish B. Narnaware. 2013. *International Journal of Computer &*

Communication Engineering Research IJCCER. *IJCCER*. Vol. 1. [s.n.]. http://www.ijccer.org/index.php/ojs/article/view/20.
Karunamurthy, Rajesh, Ferhat Khendek, and Roch H. Glitho. 2012. "A Novel Architecture for Web Service Composition." *Journal of Network and Computer Applications* 35 (2): 787–802. doi:10.1016/j.jnca.2011.11.012.
Klusch, Matthias, and Andreas Gerber. 2006. "Fast Composition Planning of OWL-S Services and Application." In *2006 European Conference on Web Services (ECOWS'06)*, 181–90. IEEE. doi:10.1109/ECOWS.2006.20.
Kodialam, M., and T.V. Lakshman. 2000. "Minimum Interference Routing with Applications to MPLS Traffic Engineering." In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, 2:884–93. IEEE. doi:10.1109/INFCOM.2000.832263.
Koomey, Jonathan G. 2007. "ESTIMATING TOTAL POWER CONSUMPTION BY SERVERS IN THE U.S. AND THE WORLD." http://www.koomey.com.
Kubernetes. 2017. "Kubernetes - Production-Grade Container Orchestration." Accessed March 15. https://kubernetes.io/.
Kumar, Dilip, Bibhudatta Sahoo, Bhaskar Mondal, and Tarni Mandal. 2015. "A Genetic Algorithmic Approach for Energy Efficient Task Consolidation in Cloud Computing." *International Journal of Computer Applications* 118 (2): 1–6. doi:10.5120/20714-3066.
Kurdi, Heba, Abeer Al-Anazi, Carlene Campbell, and Auhood Al Faries. 2015. "A Combinatorial Optimization Algorithm for Multiple Cloud Service Composition." *Computers & Electrical Engineering* 42 (February): 107–13. doi:10.1016/j.compeleceng.2014.11.002.
Kusic, Dara, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. 2008. "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control." In *2008 International Conference on Autonomic Computing*, 3–12. IEEE. doi:10.1109/ICAC.2008.31.
Kvm. 2017. "KVM (Kernel-Based Virtual Machine)." Accessed March 15. https://www.linux-kvm.org/page/Main_Page.
Larumbe, Federico, and Brunilde Sansò. 2012. "Optimal Location of Data Centers and Software Components in Cloud Computing Network Design." In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, 841–44. IEEE. doi:10.1109/CCGrid.2012.124.
Laszewski, Gregor von, Lizhe Wang, Andrew J. Younge, and Xi He. 2009. "Power-Aware Scheduling of Virtual Machines in DVFS-Enabled Clusters." In *2009 IEEE International Conference on Cluster Computing and Workshops*, 1–10. IEEE. doi:10.1109/CLUSTR.2009.5289182.
Lazzeri, Francesco, Gianmarco Bruno, Jeroen Nijhof, Alessio Giorgetti, and Piero Castoldi. 2015. "Efficient Label Encoding in Segment-Routing Enabled Optical Networks." In *2015 International Conference on Optical Network Design and Modeling (ONDM)*, 34–38. IEEE. doi:10.1109/ONDM.2015.7127270.
Lecue, Freddy, and Nikolay Mehandjiev. 2011. "Seeking Quality of Web Service Composition in a Semantic Dimension." *IEEE Transactions on Knowledge and Data Engineering* 23 (6): 942–59. doi:10.1109/TKDE.2010.237.
Lee, Ming-Chieh, and Jang-Ping Sheu. 2016. "An Efficient Routing Algorithm Based on Segment Routing in Software-Defined Networking." *Computer Networks* 103 (July): 44–55. doi:10.1016/j.comnet.2016.03.017.
Lee, Young Choon, and Albert Y. Zomaya. 2012. "Energy Efficient Utilization of Resources in Cloud Computing Systems." *The Journal of Supercomputing* 60 (2). Springer US: 268–80. doi:10.1007/s11227-010-0421-3.
Lemos, Angel Lagares, Florian Daniel, and Boualem Benatallah. 2015. "Web Service Composition." *ACM Computing Surveys* 48 (3). ACM: 1–41. doi:10.1145/2831270.
Levesque, Hector, Fiora Pirri, and Ray Reiter. 1998. "Foundations for the Situation Calculus." *Computer and Information Science* 3 (3). http:/.
Li, Bo, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. 2009. "EnaCloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments." In *2009 IEEE International Conference on Cloud Computing*, 17–24. IEEE. doi:10.1109/CLOUD.2009.72.
Liang Luo, Wenjun Wu, Dichen Di, Fei Zhang, Yizhou Yan, and Yaokuan Mao. 2012. "A Resource Scheduling Algorithm of Cloud Computing Based on Energy Efficient Optimization Methods." In *2012 International Green Computing Conference (IGCC)*, 1–6. IEEE. doi:10.1109/IGCC.2012.6322251.
Limbani, Oza. 2012. "A Proposed Service Broker Policy for Data Center Selection in Cloud Environment with Implementation." *Int. J. Comput. Technol. Appl.,*.
LinuxContainers. 2017. "Linux Containers." Accessed March 15. https://linuxcontainers.org/.
Liu, Fang, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. 2011. "NIST Cloud

Computing Reference Architecture Recommendations of the National Institute of Standards and Technology." *Cloud Computing Program, Information Technology Laboratory*.
Liu, Tao, Qingrui Li, and Ping Liang. 2012. "An Energy-Balancing Clustering Approach for Gradient-Based Routing in Wireless Sensor Networks." *Computer Communications* 35 (17): 2150–61. doi:10.1016/j.comcom.2012.06.013.
Loganathan, Shyamala, and Saswati Mukherjee. 2013. "Differentiated Policy Based Job Scheduling with Queue Model and Advanced Reservation Technique in a Private Cloud Environment." Springer, Berlin, Heidelberg, 32–39. doi:10.1007/978-3-642-38027-3_4.
Lu, Junwen, Yongsheng Hao, Lina Wang, and Mai Zheng. 2015. "Towards Efficient Service Composition in Multi-Cloud Environment." In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 65–70. IEEE. doi:10.1109/CSCI.2015.69.
Luo, Jun-Zhou, Jing-Ya Zhou, and Zhi-Ang Wu. 2009. "An Adaptive Algorithm for QoS-Aware Service Composition in Grid Environments." *Service Oriented Computing and Applications* 3 (3). Springer-Verlag: 217–26. doi:10.1007/s11761-009-0047-6.
Malboubi, Mehdi, Liyuan Wang, Chen-Nee Chuah, and Puneet Sharma. 2014. "Intelligent SDN Based Traffic (de)Aggregation and Measurement Paradigm (iSTAMP)." In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 934–42. IEEE. doi:10.1109/INFOCOM.2014.6848022.
Martínez, Alberto, Marta Patiño-Martínez, Ricardo Jiménez-Peris, and Francisco Pérez-Sorrosal. 2005. "ZenFlow: A Visual Web Service Composition Tool for BPEL4WS *." In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 181–88. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.6247&rep=rep1&type=pdf.
Mazzucco, Michele, Dmytro Dyachuk, and Ralph Deters. 2010. "Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies." In *2010 IEEE 3rd International Conference on Cloud Computing*, 131–38. IEEE. doi:10.1109/CLOUD.2010.68.
Mell, Peter, and Timothy Grance. 2011. "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology." doi:10.6028/NIST.SP.800-145.
MicrosoftAzure. 2017. "Microsoft Azure: Cloud Computing Platform Services." Accessed March 15. https://azure.microsoft.com/en-us/.
Ming Lu, Ye, and Vincent W. S. Wong. 2007. "An Energy-Efficient Multipath Routing Protocol for Wireless Sensor Networks." *International Journal of Communication Systems* 20 (7). John Wiley & Sons, Ltd.: 747–66. doi:10.1002/dac.843.
Mishra, Rakesh Kumar, Sandeep Kumar, and B Sreenu Naik. 2014. "Priority Based Round-Robin Service Broker Algorithm for Cloud-Analyst." In *2014 IEEE International Advance Computing Conference (IACC)*, 878–81. IEEE. doi:10.1109/IAdCC.2014.6779438.
Mitrani, Isi. 2011. "Service Center Trade-Offs between Customer Impatience and Power Consumption." *Performance Evaluation* 68 (11): 1222–31. doi:10.1016/j.peva.2011.07.017.
———. 2013. "Managing Performance and Power Consumption in a Server Farm." *Annals of Operations Research* 202 (1). Springer US: 121–34. doi:10.1007/s10479-011-0932-1.
Montoya, Germán A., and Yezid Donoso. 2013. "Energy Load Balancing Strategy to Extend Lifetime in Wireless Sensor Networks." *Procedia Computer Science* 17: 395–402. doi:10.1016/j.procs.2013.05.051.
Murtazaev, Aziz, and Sangyoon Oh. 2011. "Sercon: Server Consolidation Algorithm Using Live Migration of Virtual Machines for Green Computing." *IETE Technical Review* 28 (3): 212. doi:10.4103/0256-4602.81230.
Nacer, Hassina, and Djamil Aissani. 2014. "Semantic Web Services: Standards, Applications, Challenges and Solutions." *Journal of Network and Computer Applications* 44 (September): 134–51. doi:10.1016/j.jnca.2014.04.015.
Nair, SK, and Sakshi Porwal. 2010. "Towards Secure Cloud Bursting, Brokerage and Aggregation." ... *ECOWS), 2010 IEEE ...*, no. 257115: 1–8. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5693261.
Naone, Erica. 2009. "Conjuring Clouds - How Engineers Are Making on-Demand Computing a Reality." *MIT Technology Review*. https://www.technologyreview.com/s/413981/conjuring-clouds/.
Nathani, Amit, Sanjay Chaudhary, and Gaurav Somani. 2012. "Policy Based Resource Allocation in IaaS Cloud." *Future Generation Computer Systems* 28 (1): 94–103. doi:10.1016/j.future.2011.05.016.
Nidhi Jain, Inderveer Chana. 2012. "Cloud Load Balancing Techniques : A Step Towards Green Computing." *IJCSI International Journal of Computer Science Issues* 9 (1). https://www.researchgate.net/publication/266489231_Cloud_Load_Balancing_Techniques_A_Step_Towards_Green_Computing.
"OpenNebula." 2017. Accessed March 16. https://opennebula.org/.
OpenStack. 2017. "Home » OpenStack Open Source Cloud Computing Software." Accessed March 15.

https://www.openstack.org/.
OpenVZ. 2017. “OpenVZ Virtuozzo Containers Wiki.” Accessed March 16. https://openvz.org/Main_Page.
ORGERIE, ANNE-CECILE, and LAURENT LEFEVRE. 2011. “ERIDIS: ENERGY-EFFICIENT RESERVATION INFRASTRUCTURE FOR LARGE-SCALE DISTRIBUTED SYSTEMS.” *Parallel Processing Letters* 21 (2). World Scientific Publishing Company: 133–54. doi:10.1142/S0129626411000138.
Panarello, Carla, Alfio Lombardo, Giovanni Schembra, Luca Chiaraviglio, and Marco Mellia. 2010. “Energy Saving and Network Performance: A Trade-off Approach,” 41–50. doi:10.1145/1791314.1791321.
Pawar, Chandrashekhar S., and Rajnikant B. Wagh. 2012. “Priority Based Dynamic Resource Allocation in Cloud Computing.” *2012 International Symposium on Cloud and Services Computing*, 1–6. doi:10.1109/ISCOS.2012.14.
Perkins, Charles E.,Elizabeth M. Belding-Royer, Samir R. Das. 2003. “Ad Hoc On-Demand Distance Vector (AODV) Routing.” *Network Working Group*. https://tools.ietf.org/html/rfc3561.
Pi, Bingfeng, Gang Zou, Chaoliang Zhong, Jun Zhang, Hao Yu, and Akihiko Matsuo. 2012. “Flow Editor: Semantic Web Service Composition Tool.” In *2012 IEEE Ninth International Conference on Services Computing*, 666–67. IEEE. doi:10.1109/SCC.2012.48.
Pinheiro, Eduardo, Ricardo Bianchini, Enrique V Carrera, and Taliver Heath. 2001. “Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems.” *Proceedings of theWorkshop on Compilers and Operating Systems for Low Power*, 182–95. http://www2.ic.uff.br/~julius/stre/pinheiro01load.pdf.
Proxmox. 2017. “Proxmox - Open Source Server Solutions.” Accessed March 16. https://www.proxmox.com/en/.
Qi, Zheng. 2006. “Load Balancing Algorithm Based on Dynamic Feedback.” *Computer Age*, 49–51.
Quan, Dang Minh, Robert Basmadjian, Hermann De Meer, Ricardo Lent, Toktam Mahmoodi, Domenico Sannelli, Federico Mezza, Luigi Telesca, and Corenten Dupont. 2011. “Energy Efficient Resource Allocation Strategy for Cloud Data Centres.” In *Computer and Information Sciences II*, 133–41. London: Springer London. doi:10.1007/978-1-4471-2155-8_16.
Randles, Martin, David Lamb, and A. Taleb-Bendiab. 2010. “A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing.” In *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, 551–56. IEEE. doi:10.1109/WAINA.2010.85.
Rani, Pushpi, Reena Chauhan, and Ritu Chauhan. 2015. “An Enhancement in Service Broker Policy for Cloud-Analyst.” *International Journal of Computer Applications* 115 (12): 975–8887. http://research.ijcaonline.org/volume115/number12/pxc3902450.pdf.
Rekha P.M., and M. Dakshayini. 2014. “Cost Based Data Center Selection Policy for Large Scale Networks.” In *2014 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, 18–23. IEEE. doi:10.1109/ICCPEIC.2014.6915333.
Ren, Xiaona, Rongheng Lin, and Hua Zou. 2011. “A Dynamic Load Balancing Strategy for Cloud Computing Platform Based on Exponential Smoothing Forecast.” In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 220–24. IEEE. doi:10.1109/CCIS.2011.6045063.
Rodriguez-Mier, Pablo, Manuel Mucientes, and Manuel Lama. 2012. “A Dynamic QoS-Aware Semantic Web Service Composition Algorithm.” In , 623–30. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-34321-6_48.
Rodriguez-Mier, Pablo, Manuel Mucientes, Juan C. Vidal, and Manuel Lama. 2012. “An Optimal and Complete Algorithm for Automatic Web Service Composition.” *International Journal of Web Services Research* 9 (2). IGI Global: 1–20. doi:10.4018/jwsr.2012040101.
Rogers, Owen, and Dave Cliff. 2012. “A Financial Brokerage Model for Cloud Computing.” *Journal of Cloud Computing: Advances, Systems and Applications* 1 (1). Springer Open Ltd: 2. doi:10.1186/2192-113X-1-2.
S. Ponnekanti, A. Fox. 2002. “SWORD: A Developer Toolkit for Web Service Composition.” In *Proceedings of the 11th International WWW Conference (WWW2002),,*. Honolulu. https://www.bibsonomy.org/bibtex/2acd4b46d0d98e21eaac9337a2b06983b/hennig.
Sarfaraz Ahmed, A. 2012. “Enhanced Proximity-Based Routing Policy for Service Brokering in Cloud Computing.” *International Journal of Engineering Research and Applications* 2 (2): 1453–55. http://www.ijera.com/papers/Vol2_issue2/IM2214531455.pdf.
Scheepers, Mathijs Jeroen. 2014. “Virtualization and Containerization of Application Infrastructure: A Comparison.” In *21st University of Twente Conference on IT*. http://referaat.cs.utwente.nl/conference/21/paper/7449/virtualization-and-containerization-of-application-infrastructure-a-comparison.pdf.
Semwal, Ashwin, and Pradeep Singh Rawat. 2014. “Performance Evaluation of Cloud Application with

Constant Data Center Configuration and Variable Service Broker Policy Using CloudSim." *International Journal of Enhanced Research in Science Technology & Engineering* 3 (1): 2319–74631. https://pdfs.semanticscholar.org/f08a/4e08488fdb8f46a77eed301d48dbefc13e54.pdf?_ga=2.15382582.1475734039.1503066130-1030030070.1503066130.

Sgambelluri, Andrea, Alessio Giorgetti, Filippo Cugini, Gianmarco Bruno, Francesco Lazzeri, and Piero Castoldi. 2015. "First Demonstration of SDN-Based Segment Routing in Multi-Layer Networks." In *Optical Fiber Communication Conference*, Th1A.5. Washington, D.C.: OSA. doi:10.1364/OFC.2015.Th1A.5.

Sharma. 2014. "Efficient Data Center Selection Policy for Service Proximity Service Broker in CloudAnalyst." *Int. J. Innovative Comp. Sci. Eng. (IJICSE)* 1 (1): 21–28.

SolarisContainers. 2017. "Solaris Containers - Oracle." Accessed March 16. http://www.oracle.com/technetwork/ server-storage/solaris/containers-169727.html.

Soltesz, Stephen, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, Larry Peterson, Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. 2007. "Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors." In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 - EuroSys '07*, 41:275. New York, New York, USA: ACM Press. doi:10.1145/1272996.1273025.

Song, Weijia, Zhen Xiao, Qi Chen, and Haipeng Luo. 2014. "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing." *IEEE Transactions on Computers* 63 (11): 2647–60. doi:10.1109/TC.2013.148.

Song, Ying, Hui Wang, Yaqiong Li, Binquan Feng, and Yuzhong Sun. 2009. "Multi-Tiered On-Demand Resource Scheduling for VM-Based Data Center." In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 148–55. IEEE. doi:10.1109/CCGRID.2009.11.

Sotomayor, Borja, Kate Keahey, and Ian Foster. 2008. "Combining Batch Execution and Leasing Using Virtual Machines." In *Proceedings of the 17th International Symposium on High Performance Distributed Computing - HPDC '08*, 87. New York, New York, USA: ACM Press. doi:10.1145/1383422.1383434.

Srikantaiah, Shekhar, Aman Kansal, and Feng Zhao. 2008. "Energy Aware Consolidation for Cloud Computing." *Proceedings of the 2008 Conference on Power Aware Computing and Systems*. USENIX Association. http://dl.acm.org/citation.cfm?id=1855620.

Sun, Le, Hai Dong, and Jamshaid Ashraf. 2012. "Survey of Service Description Languages and Their Issues in Cloud Computing." In *2012 Eighth International Conference on Semantics, Knowledge and Grids*, 128–35. IEEE. doi:10.1109/SKG.2012.49.

Takeda, Shingo, and Toshinori Takemura. 2010. "A Rank-Based VM Consolidation Method for Power Saving in Datacenters." *Information and Media Technologies IPSJ Transactions on Advanced Computing Systems* 5 (32): 994–1002. https://www.jstage.jst.go.jp/article/imt/5/3/5_3_994/_pdf.

Taleb, Tarik, Marius Corici, Carlos Parada, Almerima Jamakovic, Simone Ruffino, Georgios Karagiannis, and Thomas Magedanz. 2015. "EASE: EPC as a Service to Ease Mobile Core Network Deployment over Cloud." *IEEE Network* 29 (2): 78–88. doi:10.1109/MNET.2015.7064907.

Tian Shaoliang, Zuo Ming, and Wu Shaowei. 2007. "An Improved Load Balancing Algorithm Based on Dynamic Feedback." *Computer Engineering and Design* 28: 572–73.

Tootoonchian, Amin, Monia Ghobadi, and Yashar Ganjali. 2010. "OpenTM: Traffic Matrix Estimator for OpenFlow Networks." http://www.pam2010.ethz.ch/papers/full-length/21.pdf.

Travostino, Franco, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunath, and Phil Yonghui Wang. 2006. "Seamless Live Migration of Virtual Machines over the MAN/WAN." *Future Generation Computer Systems* 22 (8): 901–7. doi:10.1016/j.future.2006.03.007.

Uchechukwu, Awada, Keqiu Li, and Yanming Shen. 2012. "Improving Cloud Computing Energy Efficiency." In *IEEE Asia Pacific Cloud Computing Congress*, 53–58. doi:10.1109/APCloudCC.2012.6486511.

Victories, Victor. 2015. "4 Types of Cloud Computing Deployment Model You Need to Know." *IBM developerWorks, Internet and Technology Blog*. https://www.ibm.com/developerworks/community/blogs/722f6200-f4ca-4eb3-9d64-8d2b58b2d4e8/entry/4_Types_of_Cloud_Computing_Deployment_Model_You_Need_to_Know1?lang=en.

VirtuozzoContainers. 2017. "Virtuozzo - Containers, VMs, Storage Virtualization." Accessed March 16. https://virtuozzo.com/.

VMware. 2017. "VMware Virtualization for Desktop & Server, Application, Public & Hybrid Clouds." Accessed March 15. http://www.vmware.com/.

Wajid, Usman, Cesar A. Marin, and Anthony Karageorgos. 2013. "Optimizing Energy Efficiency in the Cloud

Using Service Composition and Runtime Adaptation Techniques." In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 115–20. IEEE. doi:10.1109/SMC.2013.27.

Wang, Jinhai, Chuanhe Huang, Kai He, Xiaomao Wang, Xi Chen, and Kuangyu Qin. 2013. "An Energy-Aware Resource Allocation Heuristics for VM Scheduling in Cloud." In *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, 587–94. IEEE. doi:10.1109/HPCC.and.EUC.2013.89.

Wang, Xiumin, Jianping Wang, Zeyu Zheng, Yinlong Xu, and Mei Yang. 2009. "Service Composition in Service-Oriented Wireless Sensor Networks with Persistent Queries." In *2009 6th IEEE Consumer Communications and Networking Conference*, 1–5. IEEE. doi:10.1109/CCNC.2009.4784868.

Whitney, Josh, and Pierre Delforge. 2014. "Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers." https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf.

Wickremasinghe, Bhathiya, Rodrigo N. Calheiros, and Rajkumar Buyya. 2010. "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications." In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 446–52. IEEE. doi:10.1109/AINA.2010.32.

Wolke, Andreas, Boldbaatar Tsend-Ayush, Carl Pfeiffer, and Martin Bichler. 2015. "More than Bin Packing: Dynamic Resource Allocation Strategies in Cloud Data Centers." *Information Systems* 52: 83–95. doi:10.1016/j.is.2015.03.003.

Xavier, M. G., M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. F. De Rose. 2013. "Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments." In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 233–40. IEEE. doi:10.1109/PDP.2013.41.

Xen. 2017. "The Xen Project, a Powerful Open Source Industry Standard for Virtualization." Accessed March 15. https://www.xenproject.org/.

Yamini, B., and D. Vetri Selvi. 2010. "Cloud Virtualization: A Potential Way to Reduce Global Warming." In *Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010)*, 55–57. IEEE. doi:10.1109/RSTSCC.2010.5712798.

Zaman, Sharrukh, and Daniel Grosu. 2013. "Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds." *Journal of Parallel and Distributed Computing* 73 (4): 495–508. doi:10.1016/j.jpdc.2012.12.006.

Zhang, Li, Frank Fowley, and Claus Pahl. 2014. "A Template Description Framework for Services as a Utility for Cloud Brokerage." *International Conference on Cloud Computing and Service Science*, no. Fehling. http://doras.dcu.ie/19796/.

Zhang, Qi, Lu Cheng, and Raouf Boutaba. 2010. "Cloud Computing: State-of-the-Art and Research Challenges." *Journal of Internet Services and Applications* 1 (1). Springer London: 7–18. doi:10.1007/s13174-010-0007-6.

Zhang, Qi, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L. Hellerstein. 2012. "Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments." In *Proceedings of the 9th International Conference on Autonomic Computing - ICAC '12*, 145. New York, New York, USA: ACM Press. doi:10.1145/2371536.2371562.

Zheng Wang, and J. Crowcroft. 1996. "Quality-of-Service Routing for Supporting Multimedia Applications." *IEEE Journal on Selected Areas in Communications* 14 (7): 1228–34. doi:10.1109/49.536364.

Zou, Guobing, Yixin Chen, Yang Xiang, Ruoyun Huang, and You Xu. 2010. "AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing." http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.210.2898&rep=rep1&type=pdf.

Appendix 1

1. T. Baker, B. Al-Dawsari, H. Tawfik, D. Reid, Y. Ngoko, GreeDi: An energy efficient routing algorithm for big data on cloud, Ad Hoc Networks, Volume 35, 2015, Pages 83-96, ISSN 1570-8705.

2. Aldawsari, B., Baker, T., & England, D. (2015). Trusted Energy-Efficient Cloud-Based Services Brokerage Platform. International Journal of Intelligent Computing Research (IJICR, 6(4), 630–639.

3. Aldawsari, Bandar; Baker, Thar; England, David, "Towards a Holistic Multicloud Brokerage System: Taxonomy, Survey, and Future Directions," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on , vol., no., pp.1467-1472, 26-28 Oct. 2015.

4. B. Aldawsari, T. Baker and D. England, "Towards a holistic brokerage system for multi-cloud environment," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 249-255.

5. T. Baker, M. Mackay, A. Shaheed, and B. Aldawsari, "Security-Oriented Cloud Platform for SOA-Based SCADA," in 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing , 2015, pp. 961–970.

6. Thar Baker, Muhammad Asim, Hissam Tawfik, Bandar Aldawsari, Rajkumar Buyya, An energy-aware service composition algorithm for multiple cloudbased IoT applications, Journal of Network and Computer Applications, Volume 89, 1 July 2017, Pages 96-108, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2017.03.008.

7. B. Aldawsari, T. Baker, M. Asim, H. Tawfik, Z. Maamar, R. Buyya, Cloud-SEnergy: A Bin-Packing based Multi-Cloud Service Broker for Energy Efficient Composition and Execution of Data-intensive Applications, **has been submitted** to the Computer Communications Journal.

Appendix 2

Table 18 Summary of notations used.

Notation	Meaning
i	user machine
DC_i	data centre
v	node
G	Interconnection graphs
P^i	power consumption
$C^i(v)$	capacity of a node
$E^i \subseteq V^i \times V^i$	interconnection nodes
$L^i : E^i \rightarrow \mathbb{N}$	latency between connected nodes
B^i	bandwidth
J_u	User's job
w_u	quantity of *Flops*
in_u	amount of input
ou_u	amount of output
$ET_{send}(i)$	sending a bit from the user to the data centre
$ET_{recv}(i)$	for the inverse sending
$\mu(i)$	time units for processing one flop
s	a web Service
s^i	service input
s^o	service output
s^{ec}	energy of service computation
I	request interface
G	goal interface
S	a set of candidate ws
π_B	broker composition plan
π_{CP}	a cloud provider composition plan
π'_B	a optimal composition plan
MCP	multiple cloud providers
CP	a cloud provider
EC	energy consumption
V^i	gives a list of all possible nodes available between any i and $a\ DC_i$