CLOUD SERVICES, NETWORKING, AND MANAGEMENT

Edited by

Nelson L. S. da Fonseca

Raouf Boutaba

IEEE
IEEE PRESS

IEEE Press Series on Networks and Services Management

Thomas Plevyak and Veli Sahin, *Series Editors*

WILEY

CLOUD SERVICES, NETWORKING, AND MANAGEMENT

Edited by

Nelson L. S. da Fonseca

Raouf Boutaba

Sponsored by IEEE COMMUNICATIONS SOCIETY

IEEE Press Series on Networks and Services Management

Thomas Plevyak and Veli Sahin, *Series Editors*

IEEE

IEEE Press

WILEY

Library of Congress Cataloging-in-Publication Data

Fonseca, Nelson L. S. da.
Cloud services, networking, and management / Nelson L. S. da Fonseca, Raouf Boutaba.
pages cm
ISBN 978-1-118-84594-3 (cloth)
1. Cloud computing. I. Boutaba, Raouf. II. Title.
QA76.585.F66 2015
004.67′82–dc23

2014037179

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

1

CLOUD ARCHITECTURES, NETWORKS, SERVICES, AND MANAGEMENT

Raouf Boutaba[1] and Nelson L. S. da Fonseca[2]

[1]*D.R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada*

[2]*Institute of Computing, State University of Campinas, Campinas, São Paulo, Brazil*

1.1 INTRODUCTION

With the wide availability of high-bandwidth, low-latency network connectivity, the Internet has enabled the delivery of rich services such as social networking, content delivery, and e-commerce at unprecedented scales. This technological trend has led to the development of cloud computing, a paradigm that harnesses the massive capacities of data centers to support the delivery of online services in a cost-effective manner. In a cloud computing environment, the traditional role of service providers is divided into two: *cloud providers* who own the physical data center and lease resources (e.g., virtual machines or VMs) to service providers; and *service providers* who use resources leased by cloud providers to execute applications. By leveraging the economies-of-scale of data centers, cloud computing can provide significant reduction in operational expenditure. At the same time, it also supports new applications such as big-data analytics (e.g., MapReduce [1]) that process massive volumes of data in a scalable and efficient fashion. The rise of cloud computing has made a profound impact on the development of the IT industry in recent years. While large companies like Google, Amazon, Facebook,

Cloud Services, Networking, and Management, First Edition.
Edited by Nelson L. S. da Fonseca and Raouf Boutaba.

and Microsoft have developed their own cloud platforms and technologies, many small companies are also embracing cloud computing by leveraging open-source software and deploying services in public clouds.

However, despite the wide adoption of cloud computing in the industry, the current cloud technologies are still far from unleashing their full potential. In fact, cloud computing was known as a buzzword for several years, and many IT companies were uncertain about how to make successful investment in cloud computing. Fortunately, with the significant attraction from both industry and academia, cloud computing is evolving rapidly, with advancements in almost all aspects, ranging from data center architectural design, scheduling and resource management, server and network virtualization, data storage, programming frameworks, energy management, pricing, service connectivity to security, and privacy.

The goal of this chapter is to provide a general introduction to cloud networking, services, and management. We first provide an overview of cloud computing, describing its key driving forces, characteristics and enabling technologies. Then, we focus on the different characteristics of cloud computing systems and key research challenges that are covered in the subsequent 14 chapters of this book. Specifically, the chapters delve into several topics related to cloud services, networking and management including virtualization and software-defined network technologies, intra- and inter- data center network architectures, resource, performance and energy management in the cloud, survivability, fault tolerance and security, mobile cloud computing, and cloud applications notably big data, scientific, and multimedia applications.

1.2 PART I: INTRODUCTION TO CLOUD COMPUTING

1.2.1 What Is Cloud Computing?

Despite being widely used in different contexts, a precise definition of cloud computing is rather elusive. In the past, there were dozens of attempts trying to provide an accurate yet concise definition of cloud computing [2]. However, most of the proposed definitions only focus on particular aspects of cloud computing, such as the business model and technology (e.g., virtualization) used in cloud environments. Due to lack of consensus on how to define cloud computing, for years cloud computing was considered a buzz word or a marketing hype in order to get businesses to invest more in their IT infrastructures. The National Institute of Standards and Technology (NIST) provided a relatively standard and widely accepted definition of cloud computing as follows: “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” [3]

NIST further defined five essential characteristics, three service models, and four deployment models, for cloud computing. The five essential characteristics include the following:

1. On-demand self-service, which states that a consumer (e.g., a service provider) can acquire resources based on service demand;

2. Broad network access, which states that cloud services can be accessed remotely from heterogeneous client platforms (e.g., mobile phones);
3. Resource pooling, where resources are pooled and shared by consumers in a multi-tenant fashion;
4. Rapid elasticity, which states that cloud resources can be rapidly provisioned and released with minimal human involvement;
5. Measured service, which states that resources are controlled (and possibly priced) by leveraging a metering capability (e.g., pay-per-use) that is appropriate to the type of the service.

These characteristics provide a relatively accurate picture of what cloud computing systems should look like. It should be mentioned that not every cloud computing system exhibits all five characteristics listed earlier. For example, in a private cloud, where the service provider owns the physical data center, the metering capability may not be necessary because there is no need to limit resource usage of the service unless it is reaching data center capacity limits. However, despite the definition and aforementioned characteristics, cloud computing can still be realized in a large number of ways, and hence one may argue the definition is still not precise enough. Today, cloud computing commonly refers to a computing model where services are hosted using resources in data centers and delivered to end users over the Internet. In our opinion, since cloud computing technologies are still evolving, finding the precise definition of cloud computing at the current moment may not be the right approach. Perhaps once the technologies have reached maturity, the true definition will naturally emerge.

1.2.2 Why Cloud Computing?

In this section, we present the motivation behind the development of cloud computing. We will also compare cloud computing with other parallel and distributed computing models and highlight their differences.

1.2.2.1 Key Driving Forces. There are several driving forces behind the success of cloud computing. The increasing demand for large-scale computation and big data analytics and economics are the most important ones. But other factors such as easy access to computation and storage, flexibility in resource allocations, and scalability play important roles.

Large-scale computation and big data: Recent years have witnessed the rise of Internet-scale applications. These applications range from social networks (e.g., facebook, twitter), video applications (e.g., Netflix, youtube), enterprise applications (e.g., SalesForce, Microsoft CRM) to personal applications (e.g., iCloud, Dropbox). These applications are commonly accessed by large numbers of users over the Internet. They are extremely large scale and resource intensive. Furthermore, they often have high performance requirements such as response time. Supporting these applications requires extremely large-scale infrastructures. For instance, Google has hundreds of compute clusters deployed worldwide with hundreds of thousands of servers. Another salient

characteristic is that these applications also require access to huge volumes of data. For instance, Facebook stores tens of petabytes of data and processes over a hundred terabytes per day. Scientific applications (e.g., brain image processing, astrophysics, ocean monitoring, and DNA analysis) are more and more deployed in the cloud. Cloud computing emerged in this context as a computing model designed for running large applications in a scalable and cost-efficient manner by harnessing massive resource capacities in data centers and by sharing the data center resources among applications in an on-demand fashion.

Economics: To support large-scale computation, cloud providers rely on inexpensive commodity hardware offering better scalability and performance/price ratio than supercomputers. By deploying a very large number of commodity machines, they leverage economies of scale bringing per unit cost down and allowing for incremental growth. On the other hand, cloud customers such as small and medium enterprises, which outsource their IT infrastructure to the cloud, avoid upfront infrastructure investment cost and instead benefit from a pay-as-you-go pricing and billing model. They can deploy their services in the cloud and make them quickly available to their own customers resulting in short time to market. They can start small and scale up and down their infrastructure based on their customers demand and pay based on usage.

Scalability: By harnessing huge computing and storage capabilities, cloud computing gives customers the illusion of infinite resources on demand. Customers can start small and scale up and down resources as needed.

Flexibility: Cloud computing is highly flexible. It allows customers to specify their resource requirements in terms of CPU cores, memory, storage, and networking capabilities. Customers are also offered the flexibility to customize the resources in terms of operating systems and possibly network stacks.

Easy access: Cloud resources are accessible from any device connected to the Internet. These devices can be traditional workstations and servers or less traditional devices such as smart phones, sensors, and appliances. Applications running in the cloud can be deployed or accessed from anywhere at anytime.

1.2.2.2 Relationship with Other Computing Models.

Cloud computing is not a completely new concept and has many similarities with existing distributed and parallel computing models such as Grid computing and Cluster computing. But cloud computing also has some distinguishing properties that explain why existing models are not used and justify the need for a new one. These can be explained according to two dimensions: scale and service-orientation. Both parallel computing and cloud, computing are used to solve large-scale problems often by subdividing these problems into smaller parts and carrying out the calculations concurrently on different processors. In the cloud, this is achieved using computational models such as MapReduce. However, while parallel computing relies on expensive supercomputers and massively parallel multi-processor machines, cloud computing uses cheap, easily replaceable commodity hardware. Grid computing uses supercomputers but can also use commodity hardware, all accessible through open, general-purpose protocols and interfaces, and distributed management and job scheduling middleware. Cloud computing differs from Grid computing in that

it provides high bandwidth between machines, that is more suitable for I/O-intensive applications such as log analysis, Web crawling, and big-data analytics. Cloud computing also differs from Grid computing in that resource management and job scheduling is centralized under a single administrative authority (cloud provider) and, unless this evolves differently in the future, provides no standard application programming interfaces (APIs). But perhaps the most distinguishing feature of cloud computing compared to previous computing models is its extensive reliance on virtualization technologies to allow for efficient sharing of resources while guaranteeing isolation between multiple cloud tenants. Regarding the second dimension, unlike other computing models designed for supporting applications and are mainly application-oriented, cloud computing extensively leverages service orientation providing everything (infrastructure, development platforms, software, and applications) as a service.

1.2.3 Architecture

Generally speaking, the architecture of a cloud computing environment can be divided into four layers: the hardware/datacenter layer, the infrastructure layer, the platform layer, and the application layer, as shown in Figure 1.1. We describe each of them in detail in the text that follows:

The hardware layer: This layer is responsible for managing the physical resources of the cloud, including physical servers, routers, and switches, and power, and cooling systems. In practice, the hardware layer is typically implemented in data centers. A data center usually contains thousands of servers that are organized in racks and interconnected through switches, routers, or other fabrics. Typical issues at hardware layer include hardware configuration, fault-tolerance, traffic management, and power and cooling resource management.

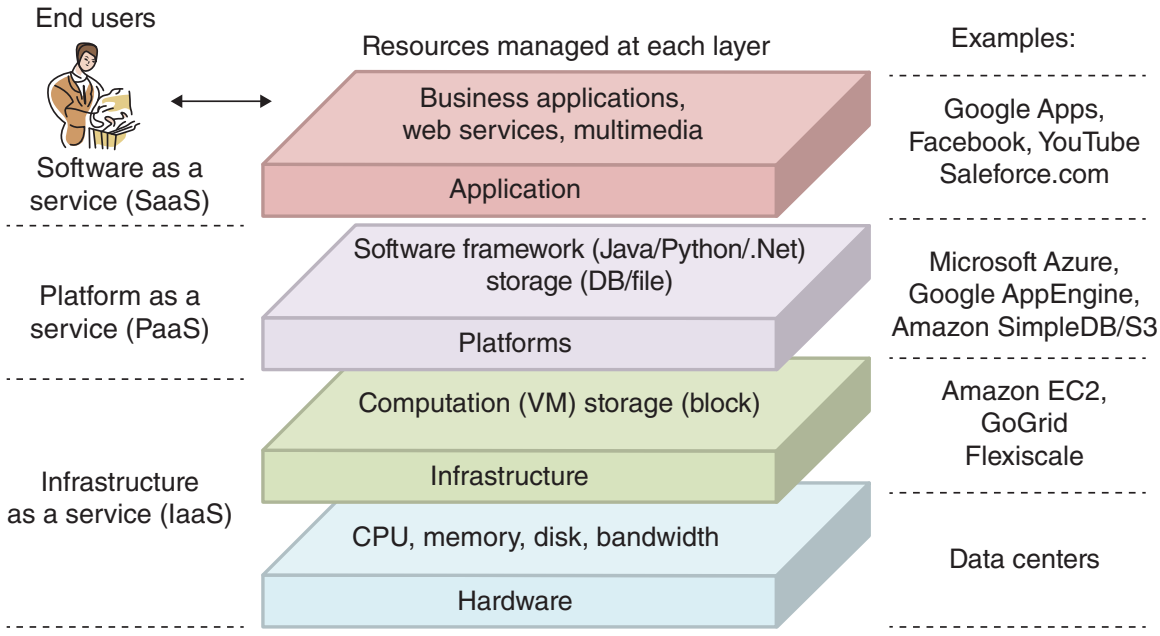

Figure 1.1. Typical architecture in a cloud computing environment.

The infrastructure layer: Also known as the virtualization layer, the infrastructure layer creates a pool of storage and computing resources by partitioning the physical resources using virtualization technologies such as Xen [4], KVM [5], and VMware [6]. The infrastructure layer is an essential component of cloud computing, since many key features, such as dynamic resource assignment, are only made available through virtualization technologies.

The platform layer: Built on top of the infrastructure layer, the platform layer consists of operating systems and application frameworks. The purpose of the platform layer is to minimize the burden of deploying applications directly into VM containers. For example, Google App Engine operates at the platform layer to provide API support for implementing storage, database, and business logic of typical Web applications.

The application layer: At the highest level of the hierarchy, the application layer consists of the actual cloud applications. Different from traditional applications, cloud applications can leverage the automatic-scaling feature to achieve better performance, availability, and lower operating cost. Compared to traditional service hosting environments such as dedicated server farms, the architecture of cloud computing is more modular. Each layer is loosely coupled with the layers above and below, allowing each layer to evolve separately. This is similar to the design of the protocol stack model for network protocols. The architectural modularity allows cloud computing to support a wide range of application requirements while reducing management and maintenance overhead.

1.2.4 Cloud Services

Cloud computing employs a service-driven business model. In other words, hardware and platform-level resources are provided as services on an on-demand basis. Conceptually, every layer of the architecture described in the previous section can be implemented as a service to the layer above. Conversely, every layer can be perceived as a customer of the layer below. However, in practice, clouds offer services that can be grouped into three categories: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

1. *Infrastructure as a service*: IaaS refers to on-demand provisioning of infrastructural resources, usually in terms of VMs. The cloud owner who offers IaaS is called an IaaS provider.
2. *Platform as a service*: PaaS refers to providing platform layer resources, including operating system support and software development frameworks.
3. *Software as a service*: SaaS refers to providing on-demand applications over the Internet.

The business model of cloud computing is depicted in Figure 1.2. According to the layered architecture of cloud computing, it is entirely possible that a PaaS provider runs its cloud on top of an IaaS providers cloud. However, in the current practice, IaaS and

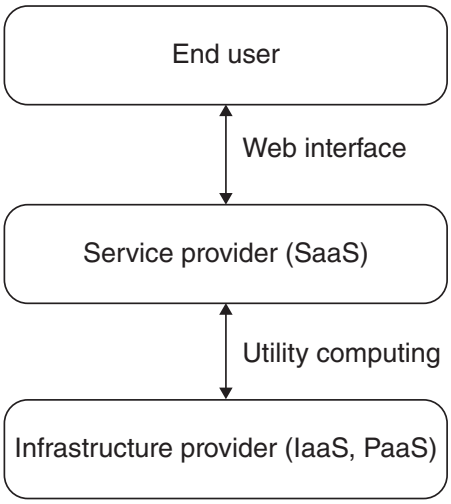

Figure 1.2. Cloud computing business model.

PaaS providers are often parts of the same organization (e.g., Google). This is why PaaS and IaaS providers are often called cloud providers [7].

1.2.4.1 Type of Clouds. There are many issues to consider when moving an enterprise application to the cloud environment. For example, some enterprises are mostly interested in lowering operation cost, while others may prefer high reliability and security. Accordingly, there are different types of clouds, each with its own benefits and drawbacks:

- *Public clouds*: A cloud in which cloud providers offer their resources as services to the general public. Public clouds offer several key benefits to service providers, including no initial capital investment on infrastructure and shifting of risks to cloud providers. However, current public cloud services still lack fine-grained control over data, network and security settings, which hampers their effectiveness in many business scenarios.
- *Private clouds*: Also known as internal clouds, private clouds are designed for exclusive use by a single organization. A private cloud may be built and managed by the organization or by external providers. A private cloud offers the highest degree of control over performance, reliability, and security. However, they are often criticized for being similar to traditional proprietary server farms and do not provide benefits such as no up-front capital costs.
- *Hybrid clouds*: A hybrid cloud is a combination of public and private cloud models that tries to address the limitations of each approach. In a hybrid cloud, part of the service infrastructure runs in private clouds while the remaining part runs in public clouds. Hybrid clouds offer more flexibility than both public and private clouds. Specifically, they provide tighter control and security over application data compared to public clouds, while still facilitating on-demand service expansion

and contraction. On the down side, designing a hybrid cloud requires carefully determining the best split between public and private cloud components.

- *Community clouds*: A community cloud refers to a cloud infrastructure that is shared between multiple organizations that have common interests or concerns. Community clouds are a specific type of cloud that relies on the common interest and limited participants to achieve efficient, reliable, and secure design of the cloud infrastructure.

Private cloud has always been the most popular type of cloud. Indeed, the development of cloud computing was largely due to the need of building data centers for hosting large-scale online services owned by large private companies, such as Amazon and Google. Subsequently, realizing the cloud infrastructure can be leased to other companies for profits, these companies have developed public cloud services. This development has also led to the creation of hybrid clouds and Community clouds, which represent different alternatives to share cloud resources among service providers. In the future, it is believed that private cloud will remain to be the dominant cloud computing model. This is because as online services continue to grow in scale and complexity, it becomes increasingly beneficial to build private cloud infrastructure to host these services. In this case, private clouds not only provide better performance and manageability than public clouds but also reduced operation cost. As the initial capital investment on a private cloud can be amortized across large number of machines over many years, in the long-term private cloud typically has lower operational cost compared to public clouds.

1.2.4.2 SME's Survey on Cloud Computing. The European Network and Information Security Agency (ENISA) has conducted a survey on the adaption of the cloud computing model by small to medium enterprises (SMEs). The survey provides an excellent overview of the benefits and limitations of today's cloud technologies. In particular, the survey has found that the main reason for adopting cloud computing is to reduce total capital expenditure on software and hardware resources. Furthermore, most of the enterprises prefer a mixture of cloud computing models (public cloud, private cloud), which comes with no surprise as each type of cloud has own benefits and limitations. Regarding the type of cloud services, it seems that IaaS, PaaS, and SaaS all received similar scores, even though SaaS is slightly in favor compared to the other two. Last, it seems that data availability, privacy, and confidentiality are the main concerns of all the surveyed enterprises. As a result, it is not surprising to see that most of the enterprises prefer to have a disaster recovery plan when considering migration to the cloud. Based on these observations, cloud providers should focus more on improving the security and reliability aspect of cloud infrastructures, as they represent the main obstacles for adopting the cloud computing model by today's enterprises.

1.2.5 Enabling Technologies

The success of cloud computing is largely driven by successful deployment of its enabling technologies. In this section, we provide an overview of cloud enabling technologies and describe how they contribute to the development of cloud computing.

1.2.5.1 Data Center Virtualization. One of the main characteristics of cloud computing is that the infrastructure (e.g., data centers) is often shared by multiple tenants (e.g., service providers) running applications with different resource requirements and performance objectives. This raises the question regarding how data center resources should be allocated and managed by each service provider. A naive solution that has been implemented in the early days is to allocate dedicated servers for each application. While this "bare-metal" strategy certainly worked in many scenarios, it also introduced many inefficiencies. In particular, if the server resource is not fully utilized by the application running on the server, the resource is wasted as no other application has the right to acquire the resource for its own execution. Motivated by this observation, the industry has adopted virtualization in today's cloud data centers. Generally speaking, virtualization aims at partitioning physical resources into virtual resources that can be allocated to applications in a flexible manner. For instance, server virtualization is a technology that partitions the physical machine into multiple VMs, each capable of running applications just like a physical machine. By separating logical resources from the underlying physical resources, server virtualization enables flexible assignment of workloads to physical machines. This not only allows workload running on multiple VMs to be consolidated on a single physical machine, but also enables a technique called VM migration, which is the process of dynamically moving a VM from one physical machine to another. Today, virtualization technologies have been widely used by cloud providers such as Amazon EC2, Rackspace, and GoGrid. By consolidating workload using fewer machines, server virtualization can deliver higher resource utilization and lower energy consumption compared to allocating dedicated servers for each application.

Another type of data center virtualization that has been largely overlooked in the past is network virtualization. Cloud applications today are becoming increasingly data-intensive. As a result, there is a pressing need to determine how data center networks should be shared by multiple tenants with diverse performance, security and manageability requirements. Motivated by these limitations, there is an emerging trend towards virtualizing data center networks in addition to server virtualization. Similar to server virtualization, network virtualization aims at creating multiple VNs on top of a shared physical network substrate allowing each VN to be implemented and managed independently. By separating logical networks from the underlying physical network, it is possible to implement network resource guarantee and introduce customized network protocols, security, and management policies. Combining with server virtualization, a fully virtualized data centers support the allocation in the form of virtual infrastructures or VIs (also known as virtual data centers (VDC)), which consist of VMs inter-connected by virtual networks. The scheduling and management of VIs have been studied extensively in recent years. Commercial cloud providers are also pushing towards this direction. For example, the Amazon Virtual Private Cloud (VPC) already provides limited features to support network virtualization in addition to server virtualization.

1.2.5.2 Cloud Networking. To ensure predictable performance over the cloud, it is of utmost importance to design efficient networks that are able to provide guaranteed performance and to scale with the ever-growing traffic volumes in the cloud. Traditional

data center network architectures suffer from many limitations that may hinder the performance of large-scale cloud services. For instance, the widely-used tree-like topology does not provide multiple paths between the nodes, and hence limits the scalability of the network and the ability to mitigate node and link congestion and failures. Moreover, current technologies like Ethernet and VLANs are not well suited to support cloud computing requirements like multi-tenancy or performance isolation between different tenants/applications. In recent years, several research works have focused on designing new data center network architectures to overcome these limitations and enhance performance, fault tolerance and scalability (e.g., VL2 [38], Portland [9], NetLord [10]). Furthermore, the advent of software-defined networking (SDN) technology brings new opportunities to redesign cloud networks [11]. Thanks to the programmability offered by this technology, it is now possible to dynamically adapt the configuration of the network based on the workload. It also makes it easy to implement policy-based network management schemes in order to achieve potential cloud providers' objectives in terms of performance, utilization, survivability, and energy efficiency.

1.2.5.3 Data Storage and Management. As mentioned previously, one of the key driving forces for cloud computing is the need to process large volumes of data in a scalable and efficient manner. As cloud data centers typically consist of commodity servers with limited storage and processing capacities, it is necessary to develop distributed storage systems that support efficient retrieval of desired data. At the same time, as failures are common in commodity machine-based data centers, the distributed storage system must also be resilient to failures. This usually implies each file block must be replicated on multiple machines. This raises challenges regarding how the distributed storage system should be designed to achieve availability and high performance, while ensuring file replicas remain consistent over time. Unfortunately, the famous CAP theorem [12] states that simultaneously achieving all three objectives (consistency, availability, and robustness to network failures) is not a viable task. As result, recently many file systems such Google File System [13], Amazon Dynamo [14], Cassandra [15] are trying to explore various trade-offs among the three objectives based on applications' needs. For example, Amazon Dynamo adopts an eventual consistency model that allow replicas to be temporary out-of-sync. By sacrificing consistency, Dynamo is able to achieve significant improvement in server response time. It is evident that these storage systems provide the foundations for building large-scale data-intensive applications that are commonly found in today's cloud data centers.

1.2.5.4 MapReduce Programming Model. Cloud computing has become the most cost-effective technology for hosting Internet-scale applications. Companies like Google and Facebook generate enormous volumes of data on a daily basis that need to be processed in a timely manner. To meet this requirement, cloud providers use computational models such as MapReduce [1] and Dryad [16]. In these models, a job spawns many small tasks that can be executed concurrently on multiple machines, resulting in significant reduction in job completion time. Furthermore, to cope with software and hardware exceptions frequent in large-scale clusters, these models provide built-in fault

tolerance features that automatically restart failed tasks when exceptions occur. As a result, these computational models are very attractive not only for running data-intensive jobs but also for computation-intensive applications. The MapReduce model, in particular, is largely used nowadays in cloud infrastructures for supporting a wide range of applications and has been adapted to several computing and cluster environments. Despite this success, the adoption of MapReduce has implications on the management of cloud workload and cluster resources, which is still largely unstudied. In particular, many challenges pertaining to MapReduce job scheduling, task and data placement, resource allocation, and sharing are yet to be addressed.

1.2.5.5 Resource Management. Resource management has always been a central theme of cloud computing. Given the large variety of applications running in the cloud, it is a challenging problem to determine how each application should be scheduled and managed in a scalable and dynamic manner. The scheduling of individual application component can be formulated as a variant of the multi-dimensional vector bin-packing problem, which is already NP-hard in the general case. Furthermore, different applications may have different scheduling needs. For example, individual tasks of a single MapReduce job can be scheduled independently over time, whereas the servers of a three-tier Web application must be scheduled simultaneously to ensure service availability. Therefore, finding a scheduling scheme that satisfy diverse application scheduling requirement is a challenging problem. The recent work on multi-framework scheduling (e.g., MESOS [17]) provides a platform to allow various scheduling frameworks, such as MapReduce, Spark, and MPI to coexist in a single cloud infrastructure. The work on distributed schedulers (e.g., Omega [18] and Sparrow [19]) also aim at improving the scalability of schedulers by having multiple schedulers perform scheduling in parallel. These technologies will provide the functionality to support a wide range of workload in the cloud data center environments.

1.2.5.6 Energy Management. Data centers consume tremendous amount of energy, not only for powering up the servers and network devices, but also for cooling down these components to prevent overheating conditions. It has been reported that energy cost accounts for 15% of the average data center operation expenditure. At the same time, such large energy consumption also raises environmental concerns regarding the carbon emissions for energy generation. As a result, improving data center energy efficiency has become a primary concern for today's data center operators. A widely used metric for measuring energy efficiency of data centers is power usage effectiveness (PUE), which is computed as the ratio between the computer infrastructure usage and the total data center power usage. Even though none of the existing data centers can achieve the ideal PUE value of 1.0, many cloud data centers today have become very energy efficient with PUE less than 1.1.

There are many techniques for improving data center energy efficiency. At the infrastructure level, many cloud providers leverage nearby renewable energy source (i.e., solar and wind) to reduce energy cost and carbon footprint. At the same time, it is also possible to leverage environmental conditions (e.g., low temperature conditions) to reduce

cooling cost. For example, Facebook recently announced the construction of a cloud data center in Sweden, right on the edge of the arctic circle, mainly due to the low air temperature that can reduce cooling cost. The Net-Zero Energy Data Center developed by HP labs leverages locally generated renewable energy and workload demand management techniques to significantly reduce the energy required to operate data centers. We believe the rapid development of cloud energy management techniques will continue to push the data center energy efficiency towards the ideal PUE value of 1.0.

1.2.5.7 Security and Privacy. Security is another major concern of cloud computing. While security is not a critical concern in many private clouds, it is often a key barrier to the adoption of cloud computing in public clouds. Specifically, since service providers typically do not have access to the physical security system of data centers, they must rely on cloud providers to achieve full data security. The cloud provider, in this context, must achieve the following objectives: (1) confidentiality, for secure data access and transfer, and (2) auditability, for attesting whether security setting of applications has been tampered or not. Confidentiality is usually achieved using cryptographic protocols, whereas auditability can be achieved using remote attestation techniques. Remote attestation typically requires a trusted platform module (TPM) to generate nonforgeable system summary (i.e., system state encrypted using TPM private key) as the proof of system security. However, in a virtualized environment like the clouds, VMs can dynamically migrate from one location to another, hence directly using remote attestation is not sufficient. In this case, it is critical to build trust mechanisms at every architectural layer of the cloud. First, the hardware layer must be trusted using hardware TPM. Second, the virtualization platform must be trusted using secure VM monitors. VM migration should only be allowed if both source and destination servers are trusted. Recent work has been devoted to designing efficient protocols for trust establishment and management.

1.3 PART II: RESEARCH CHALLENGES—THE CHAPTERS IN THIS BOOK

This book covers the fundamentals of cloud services, networking and management and focuses on most prominent research challenges that have drawn the attention of the IT community in the past few years. Each of the 14 chapters of this book provides an overview of some of the key architectures, features, and technologies of cloud services, networking and management systems and highlights state-of-the-art solutions and possible research gaps. The chapters of the book are written by knowledgeable authors that were carefully selected based on their expertise in the field. Each chapter went through a rigorous review process, including external reviewers, the book editors Raouf Boutaba and Nelson Fonseca, and the series editors Tom Plevyak and Veli Sahin. In the following, we briefly describe the topics covered by the different chapters of this book.

1.3.1 Virtualization in the Cloud

Virtualization is one of the key enabling technologies that made cloud computing model a reality. Initially, virtualization technologies have allowed to partition a physical server

into multiple isolated environments called VMs that may eventually host different operating systems and be used by different users or applications. As cloud computing evolved, virtualization technologies have matured and have been extended to consider not only the partitioning of servers but also the partitioning of the networking resources (e.g., links, switches and routers). Hence, it is now possible to provide each cloud user with a VI encompassing VMs, virtual links, and virtual routers and switches. In this context, several challenges arise especially regarding the management of the resulting virtualized environment where different types of resources are shared among multiple users.

In this chapter, the authors outline the main characteristics of these virtualized infrastructures and shed light on the different management operations that need to be implemented in such environments. They then summarize the ongoing efforts towards defining open standard interfaces to support virtualization and interoperability in the cloud. Finally, the chapter provides a brief overview of the main open-source cloud management platforms that have recently emerged.

1.3.2 VM Migration

One of the powerful features brought by virtualization is the ability to easily migrate VMs within the same data center or even between geographically distributed data centers. This feature provides an unprecedented flexibility to network and data center operators allowing them to perform several management tasks like dynamically optimizing resource allocations, improving fault tolerance, consolidating workloads, avoiding server overload, and scheduling maintenance activities. Despite all these benefits, VM migration induces several costs, including higher utilization of computing and networking resources, inevitable service downtime, security risks, and more complex management challenges. As a result, a large number of migration techniques have been recently proposed in the literature in order to minimize these costs and make VM migration a more effective and secure tool in the hand of cloud providers.

This chapter starts by providing an overview of VM migration techniques. It then presents, XenFlow, a tool based on Xen and OpenFlow, and allowing to deploy, isolate and migrate VIs. Finally, the authors discuss potential security threats that can arise when using VM migration.

1.3.3 Data Center Networks and Relevant Standards

Today's cloud data centers are housing hundreds of thousands of machines that continuously need to exchange tremendous amounts of data with stringent performance requirements in terms of bandwidth, delay, jitter, and loss rate. In this context, the data center network plays a central role to ensure a reliable and efficient communication between machines, and thereby guarantee continuous operation of the data center and effective delivery of the cloud services. A data center network architecture is typically defined by the network topology (i.e., the way equipment are inter-connected) as well as the adopted switching, routing, and addressing schemes and protocols (e.g., Ethernet and IP).

Traditional data center network architectures suffer from several limitations and are not able to satisfy new application requirements spawned by cloud computing model in terms of scalability, multitenancy and performance isolation. For instance, the widely used tree-like topology does not provide multiple paths between the nodes, and hence limits the ability to survive node and link failures. Also, current switches have limited forwarding table sizes, making it difficult for traditional data center networks to handle the large number of VMs that may exist in virtualized cloud environments. Another issue is with the performance isolation between tenants as there is no bandwidth allocation mechanism in place to ensure predictable network performance for each of them.

In order to cope with these limitations, a lot of attention has been devoted in the past few years to study the performance of existing architectures and to design better solutions. This chapter dwells on these solutions covering data center network architectures, topologies, routing protocols and addressing schemes that have been recently proposed in the literature.

1.3.4 Interdata Center Networks

In recent years, cloud providers have largely relied on large-scale cloud infrastructures to support Internet-scale applications efficiently. Typically, these infrastructures are composed of several geographically distributed data centers connected through a backbone network (i.e., an inter-data center network). In this context, a key challenge facing cloud providers is to build cost-effective backbone networks while taking into account several considerations and requirements including scalability, energy efficiency, resilience, and reliability. To address this challenge, many factors should be considered. The scalability requirement is due to the fact that the volume of data exchanged between data centers is growing exponentially with the ever-increasing demand in cloud environments. The energy efficiency requirement concerns how to minimize the energy consumption of the infrastructure. Such a requirement is not only crucial to make the infrastructure more green and environmental-friendly but also essential to cut down operational expenses. Finally, the resilience of the interdata center network requirement is fundamental to maintain a continuous and reliable cloud services.

This chapter investigates the different possible alternatives to design and manage cost-efficient cloud backbones. It then presents mathematical formulations and heuristic solutions that could be adopted to achieve desired objectives in terms of energy efficiency, resilience and reliability. Finally, the authors discuss open issues and key research directions related to this topic.

1.3.5 OpenFlow and SDN for Clouds

The past few years have witnessed the rise of SDN, a technology that makes it possible to dynamically configure and program networking elements. Combined with cloud computing technologies, SDN enables the design of highly dynamic, efficient, and cost-effective shared application platforms that can support the rapid deployment of Internet applications and services.

This chapter discusses the challenges faced to integrate SDN technology in cloud application platforms. It first provides a brief overview of the fundamental concepts of SDN including OpenFlow technology and tools like Open vSwitch. It also introduces the cloud platform OpenStack with a focus on its Networking Service (i.e., Neutron project), and shows how cloud computing environments can benefit from SDN technology to provide guaranteed networking resources within a data center and to interconnect data centers. The authors also review major open source efforts that attempt to integrate SDN technology in cloud management platforms (e.g., OpenDaylight open source project) and discuss the notion of software-defined infrastructure (SDI).

1.3.6 Mobile Cloud Computing

Mobile cloud computing has recently emerged as a new paradigm that combines cloud computing with mobile network technology with the goal of putting the scalability and limitless resources of the cloud into the hands of mobile service and application providers. However, despite of its potential benefits, the growth of mobile cloud computing in recent years was hampered by several technical challenges and risks. These challenges and risks are mainly due to the inherent limitations of mobile devices such as the scarcity of resources, the limited energy supply, the intermittent connectivity in wireless networks, security risks, and legal/environmental risks.

This chapter starts by providing an overview of mobile cloud computing application models and frameworks. It also defines risk management and identifies and analyzes prevalent risk factors found in mobile cloud computing environments. The authors also present an analysis of mobile cloud frameworks from a risk management perspective and discusses the effectiveness of traditional risk approaches to address mobile cloud computing risks.

1.3.7 Resource Management and Scheduling

Resource allocation and scheduling are two crucial functions in cloud computing environments. Generally speaking, cloud providers are responsible for allocating resources (e.g., VMs) with the goal of satisfying the promised service-level agreement (SLA) while increasing their profit. This can be achieved by reducing operational costs (e.g., energy costs) and sharing resources among the different users. At the opposite side, cloud users are responsible for application scheduling that aims at mapping tasks from applications submitted by users to computational resources in the system. The goals of scheduling include maximizing the usage of the leased resources, and minimizing costs by dynamically adjusting the leased resources to the demand while maintaining the required quality of service.

Resource allocation and scheduling are both vital to cloud users and providers, but they both have their own specifics, challenges and potentially conflicting objectives. This chapter starts by a review of the different cloud types and service models and then discusses the typical objectives of cloud providers and their clients. The chapter provides also mathematical formulations to the problems, VM allocation, and application

scheduling. It surveys some of the existing solutions and discusses their strengths and weaknesses. Finally, it points out the key research directions pertaining to resource management in cloud environments.

1.3.8 Autonomic Performance Management for Multi-Clouds

The growing popularity of the cloud computing model have led to the emergence of multiclouds or clouds of clouds where multiple cloud systems are federated together to further improve and enhance cloud services. Multiclouds have several benefits that range from improving availability, to reducing lock-in, and optimizing costs beyond what can be achieved within a single cloud. At the same time, multi-clouds bring new challenges in terms of the design, development, deployment, monitoring, and management of multi-tier applications able to capitalize on the advantages of such distributed infrastructures. As a matter of fact, the responsibility for addressing these challenges is shared among cloud providers and cloud users depending on the type of service (i.e., IaaS, PaaS, and SaaS) and SLAs. For instance, from an IaaS cloud provider's perspective, management focuses mainly on maintaining the infrastructure, allocating resources requested by clients and ensuring their high availability. By contrast, cloud users are responsible for implementing, deploying and monitoring applications running on top of resources that are eventually leased from several providers. In this context, a compelling challenge that is currently attracting a lot of attention is how to develop sophisticated tools that simplify the process of deploying, managing, monitoring, and maintaining large-scale applications over multi-clouds.

This chapter focuses on this particular challenge and provides a detailed overview of the design and implementation of XCAMP, the X-Cloud Application Management Platform that allows to automate application deployment and management in multitier clouds. It also highlights key research challenges that require further investigation in the context of performance management and monitoring in distributed cloud environments.

1.3.9 Energy Management

Cloud computing environments mainly consist of data centers where thousands of servers and other systems (e.g, power distribution and cooling equipment) are consuming tremendous amounts of energy. Recent reports have revealed that energy costs represent more than 12% of the total data center operational expenditures, which translates into millions of dollars. More importantly, high energy consumption is usually synonymous of high carbon footprint, raising serious environmental concerns and pushing governments to put in place more stringent regulations to protect the environment. Consequently, reducing energy consumption has become one of the key challenges facing today's data center managers. Recently, a large body of work has been dedicated to investigate possible techniques to achieve more energy-efficient and environment-friendly infrastructures. Many solutions have been proposed including dynamic capacity provisioning and optimal usage of renewable sources of energy (e.g., wind power and solar).

This chapter further details the trends in energy management solutions in cloud data centers. It first surveys energy-aware resource scheduling and allocation schemes aiming at improving energy efficiency, and then provides a detailed description of GreenCloud, an energy-aware cloud data center simulator.

1.3.10 Survivability and Fault Tolerance in the Cloud

Despite the success of cloud computing, its widespread and full-scale adoption have been hampered by the lack of strict guarantees on the reliability and availability of the offered resources and services. Indeed, outages, failures and service disruption can be fatal for many businesses. Not only they incur significant revenue loss—as much as hundreds of thousands of dollars per minute for some services—but they may also hurt the business reputation in the long term and impact on customers' loyalty and satisfaction. Unfortunately, major cloud providers like Amazon EC2, Google, and Rackspace are not yet able to satisfy the high availability and reliability levels required for such critical services.

Consequently, a growing body of work has attempted to address this problem and to propose solutions to improve the reliability of cloud services and eventually provide more stringent guarantees to cloud users. This chapter provides a comprehensive literature survey on this particular topic. It first lays out cloud computing and survivability-related concepts, and then covers recent studies that analyzed and characterized the types of failures found in cloud environments. Subsequently, the authors survey and compare the solutions put forward to enhance cloud services' fault-tolerance and to guarantee high availability of cloud resources.

1.3.11 Cloud Security

Security has always been a key issue for cloud-based services and several solutions have been proposed to protect the cloud from malicious attacks. In particular, intrusion detection systems (IDS) and intrusion prevention systems (IPS) have been widely deployed to improve cloud security and have been recently empowered with new technologies like SDN to further enhance their effectiveness. For instance, the SDN technology has been leveraged to dynamically reconfigure the cloud network and services and better protect them from malicious traffic. In this context, this chapter introduces FlowIPS, an OpenFlow-based IPS solution for intrusion prevention in cloud environments. FlowIPS implements SDN-based control functions based on Open vSwitch (OVS) and provides novel Network Reconfiguration (NR) features by programming POX controllers. Finally, the chapter presents the performance evaluation of FlowIPS that demonstrates its efficiency compared to traditional IPS solutions.

1.3.12 Big Data on Clouds

Big data has emerged as a new term that describes all challenges related to the manipulation of large amounts of data including data collection, storage, processing, analysis, and visualization.

This chapter articulates some of the success enablers for deploying Big Data on Clouds (BDOC). It starts by providing some historical perspectives and by describing emerging Internet services and applications. It then describes some legal issues related to big data on clouds. In particular, it highlights emerging hybrid big data management roles, the development and operations (DevOps), and Site Reliability Engineering (SRE). Finally, the chapter discusses science, technology, engineering, and mathematics (STEM) talent cultivation and engagement, as an enabler to technical succession and future success for global enterprises of big data on clouds.

1.3.13 Scientific Applications on Clouds

In order to cope with the requirements of scientific applications, cloud providers have recently proposed new coordination and management tools and services (e.g., Amazon Simple WorkFlow or SWF) in order to automate and streamline task processes executed by the cloud applications. Such services allow to specify the dependencies between the tasks, their order of execution and make it possible to track their progress and the current state of each of them. In this context, a compelling challenge is to ensure the compatibility between existing workflow systems and to provide the possibility to reuse scientific legacy code.

This chapter presents a software engineering solution that allows the scientific workflow community to use Amazon cloud via a single front-end converter. In particular, it describes a wrapper service for executing legacy code using Amazon SWF. The chapter also describes the experimental results demonstrating that the automatically SWF application generated by the wrapper provides a performance comparable to the native manually optimized workflow.

1.3.14 Interactive Multimedia Applications on Clouds

The booming popularity of cloud computing has led to the emergence of a large array of new applications such as social networking, gaming, live streaming, TV broadcasting, and content delivery. For instance, cloud gaming allows direct on-demand access to games whose content is stored in the cloud and streamed directly to end users through thin clients. As a result, less powerful game consoles or computers are needed as most of the processing is carried out in the hosting cloud, leveraging its seemingly unlimited resources. Another prominent cloud application is the Massive user-generated content (UGC) live streaming that allows each simple Internet user to become a TV or content provider. A similar application that has become extremely popular is time-shifting on-demand TV as many services like catch-up TV (i.e., the content of a TV channel is recorded for many days and can be requested on demand) and TV surfing (i.e., the possibility of pausing, forwarding or rewinding of a video stream) have recently became widely demanded. Naturally, the cloud is the ideal platform to host such services as it provides the processing and storage capacity required to ensure a high quality of service. However, several challenges are not addressed yet especially because of the stringent performance requirements (e.g., delay) of such multimedia applications and the increasing amounts of traffic they generate.

This chapter discusses the deployment of these applications over the cloud. It starts by laying out content delivery models in general, and then provides a detailed study of the performance of three prominent multimedia cloud applications, namely cloud gaming, massive user-generated content live streaming and time-shifting on-Demand TV.

1.4 CONCLUSION

Editing and preparing a book on such an important topic is a challenging task requiring a lot of effort and time. As the editors of this book, we are grateful to many individuals who contributed to its successful completion. We would like to thank the chapters' authors for their high-quality contributions, the reviewers for their insightful comments and feedback, and the book series editors for their support and guidance. Finally, we hope that the reader finds the topics and the discussions presented in this book informative, interesting, and inspiring and pave the way for designing new cloud platforms able to meet the requirements of future Internet applications and services.

REFERENCES

1. J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
2. L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.
3. P. Mell and T. Grance, "The NIST definition of cloud computing (draft)," *NIST Special Publication*, vol. 800, no. 145, p. 7, 2011.
4. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.
5. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: The linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, Dttawa, Dntorio, Canada, 2007, pp. 225–230.
6. F. Guthrie, S. Lowe, and K. Coleman, *VMware vSphere Design*. John Wiley & Sons, Indianapolis, IN, 2013.
7. A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Department of Electrical Engineering and Computer Sciences*, *University of California, Berkeley, CA, Rep. UCB/EECS*, vol. 28, p. 13, 2009.
8. A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proceedings ACM SIGCOMM*, Barcelona, Spain, August 2009.
9. R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proceedings ACM SIGCOMM*, Barcelona, Spain, August 2009.

10. J. Mudigonda, P. Yalagandula, B. Stiekes, and Y. Pouffary, "NetLord: A scalable multi-tenant network architecture for virtualized datacenters," in *Proceedings ACM SIGCOMM*, Toronto, Dntorio, Canada, August 2011.
11. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Computer Communnication Review*, vol. 38, no. 2, pp. 69–74, March 2008.
12. S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.
13. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.
14. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.
15. A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.
16. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 59–72, 2007.
17. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "MESOS: A platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, Boston, MA, 2011, pp. 22–22.
18. M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proceedings of the 8th ACM European Conference on Computer Systems*, Prague, Czech Republic. ACM, New York, 2013, pp. 351–364.
19. K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, Farmington, PA. ACM, New York, 2013, pp. 69–84.

4

DATACENTER NETWORKS AND RELEVANT STANDARDS

Daniel S. Marcon, Rodrigo R. Oliveira, Luciano P. Gaspary, and Marinho P. Barcellos

Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

4.1 OVERVIEW

Datacenters are the core of cloud computing, and their network is an essential component to allow distributed applications to run efficiently and predictably [1]. However, not all datacenters provide cloud computing. In fact, there are two main types of datacenters: production and cloud. Production datacenters are often shared by one tenant or among multiple (possibly competing) groups, services, and applications, but with low rate of arrival and departure. They run data analytics jobs with relatively little variation in demands, and their size varies from hundreds of servers to tens of thousands of servers. Cloud datacenters, in contrast, have high rate of tenant arrival and departure (churn) [2], run both user-facing applications and inward computation, require elasticity (since application demands are highly variable), and consist of tens to hundreds of thousands of physical servers [3]. Moreover, clouds can comprise several datacenters spread around the world. As an example, Google, Microsoft, and Amazon (three of the biggest players in the market) have datacenters in four continents; and each company has over 900,000 servers.

Cloud Services, Networking, and Management, First Edition.
Edited by Nelson L. S. da Fonseca and Raouf Boutaba.

This chapter presents an in-depth study of datacenter networks (DCNs), relevant standards, and operation. Our goal here is three-fold: (i) provide a detailed view of the networking infrastructure connecting the set of servers of the datacenter via high-speed links and commodity off-the-shelf (COTS) switches [4]; (ii) discuss the addressing and routing mechanisms employed in this kind of network; and (iii) show how the nature of traffic may impact DCNs and affect design decisions.

Providers typically have three main goals when designing a DCN [5]: scalability, fault tolerance, and agility. First, the infrastructure must scale to a large number of servers (and preferably allow incremental expansion with commodity equipment and little effort). Second, a DCN should be fault tolerant against failures of both computing and network resources. Third, a DCN ideally needs to be agile enough to assign any virtual machine or, in short, VM (which is part of a service or application) to any server [6]. As a matter of fact, DCNs should ensure that computations are not bottlenecked on communication [7].

Currently, providers attempt to meet these goals by implementing the network as a multi-rooted tree [1], using LAN technology for VM addressing and two main strategies for routing: equal-cost multipath (ECMP) and valiant load balancing (VLB). The shared nature of DCNs among a myriad of applications and tenants and high scalability requirements, however, introduce several challenges for architecture design, protocols and strategies employed inside the network. Furthermore, the type of traffic in DCNs is significantly different from traditional networks [8]. Therefore, we also survey recent proposals in the literature to address the limitations of technologies used in today's DCNs.

We structure this chapter as follows. First, we begin by examining the typical multi-rooted tree topology used in current datacenters and discuss its benefits and drawbacks. Then, we take a look at novel topologies proposed in the literature, and how network expansion can be performed in a cost-efficient way for providers. After addressing the structure of the network, we look into the traffic characteristics of these high-performance, dynamic networks and discuss proposals for traffic management on top of existing topologies. Based on the aspects discussed so far, we present layer-2 and layer-3 routing, its requirements and strategies typically employed to perform such task. We also examine existing mechanisms used for VM addressing in the cloud platform and novel proposals to increase flexibility and isolation for tenants. Finally, we discuss the most relevant open research challenges and close this chapter with a brief summary of DCNs.

4.2 TOPOLOGIES

In this section, we present an overview of datacenter topologies. The topology describes how devices (routers, switches and servers) are interconnected. More formally, this is represented as a graph, in which switches, routers and servers are the nodes, and links are the edges.

4.2.1 Typical Topology

Figure 4.1 shows a canonical three-tiered multi-rooted tree-like physical topology, which is implemented in current datacenters [1, 9]. The three tiers are: (1) the access (edge) layer, comprising the top-of-rack (ToR) switches that connect servers mounted on every rack; (2) the aggregation (distribution) layer, consisting of devices that interconnect ToR switches in the access layer; and (3) the core layer, formed by routers that interconnect switches in the aggregation layer. Furthermore, every ToR switch may be connected to multiple aggregation switches for redundancy (usually 1+1 redundancy) and every aggregation switch is connected to multiple core switches. Typically, a three-tiered network is implemented in datacenters with more than 8000 servers [4]. In smaller datacenters, the core and aggregation layers are collapsed into one tier, resulting in a two-tiered datacenter topology (flat layer-2 topology) [9].

This multitiered topology has a significant amount of oversubscription, where servers attached to ToR switches have significantly more (possibly an order of magnitude) provisioned bandwidth between one another than they do with hosts in other racks [3]. Providers employ this technique in order to reduce costs and improve resource utilization, which are key properties to help them achieve economies of scale.

This topology, however, presents some drawbacks. First, the limited bisection bandwidth[1] constrains server-to-server capacity, and resources eventually get fragmented (limiting agility) [11, 12]. Second, multiple paths are poorly exploited (e.g., only a single path is used within a layer-2 domain by spanning tree protocol), which may potentially cause congestion on some links even though other paths exist in the network and have available capacity. Third, the rigid structure hinders incremental expansion [13].

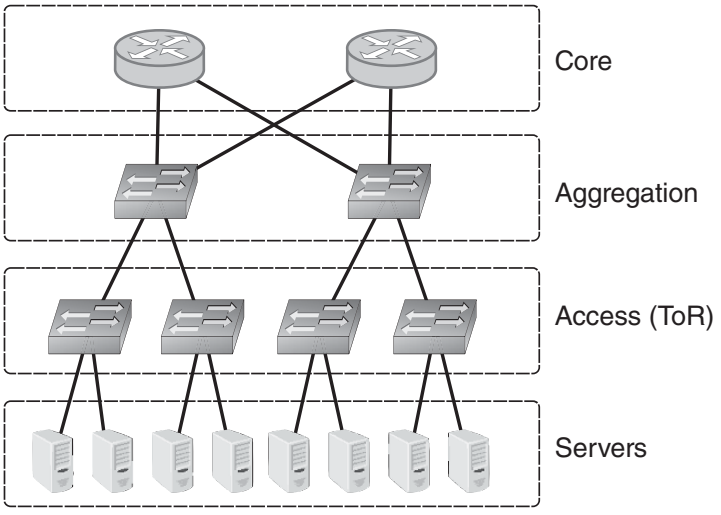

Figure 4.1. A canonical three-tiered tree-like datacenter network topology.

[1]The bisection bandwidth of the network is the worst-case segmentation (i.e., with minimum bandwidth) of the network in two equally-sized partitions [10].

Fourth, the topology is inherently failure-prone due to the use of many links, switches and servers [14]. To address these limitations, novel network architectures have been recently proposed; they can be organized in three classes [15]: switch-oriented, hybrid switch/server and server-only topologies.

4.2.2 Switch-Oriented Topologies

These proposals use commodity switches to perform routing functions, and follow a clos-based design or leverage runtime reconfigurable optical devices. A clos network [16] consists of multiple layers of switches; each switch in a layer is connected to all switches in the previous and next layers, which provides path diversity and graceful bandwidth degradation in case of failures. Two proposals follow the Clos design: VL2 [6] and Fat-Tree [4]. VL2, shown in Figure 4.2a, is an architecture for large-scale datacenters and provides multiple uniform paths between servers and full bisection bandwidth (i.e., it is non-oversubscribed). Fat-Tree, in turn, is a folded Clos topology. The topology, shown in Figure 4.2b, is organized in a non-oversubscribed k-ary tree-like structure, consisting of k-port switches. There are k two-layer pods with $k/2$ switches. Each $k/2$ switch in the lower layer is connected to $k/2$ servers, and the remaining ports are connected to $k/2$ aggregation switches. Each of the $(k/2)^2 k$-port core switches has one port connected to each of k pods. In general, a fat-tree built with k-port switches supports $k^3/4$ hosts. Despite the high capacity offered (agility is guaranteed), these architectures increase wiring costs (because of the number of links).

Optical switching architecture (OSA) [17], in turn, uses runtime reconfigurable optical devices to dynamically change physical topology and one-hop link capacities (within 10 milliseconds). It employs hop-by-hop stitching of multiple optical links to provide all-to-all connectivity for the highly dynamic and variable network demands of cloud applications. This method is shown in the example of Figure 4.3. Suppose that demands change from the left table to the right table in the figure (with a new highlighted entry). The topology must be adapted to the new traffic pattern, otherwise there will be at least one congested link. One possible approach is to increase capacity of link F–G (by reducing capacity of links F–D and G–C), so congestion can be avoided. Despite the flexibility achieved, OSA suffers from scalability issues, since it is designed to connect only a few

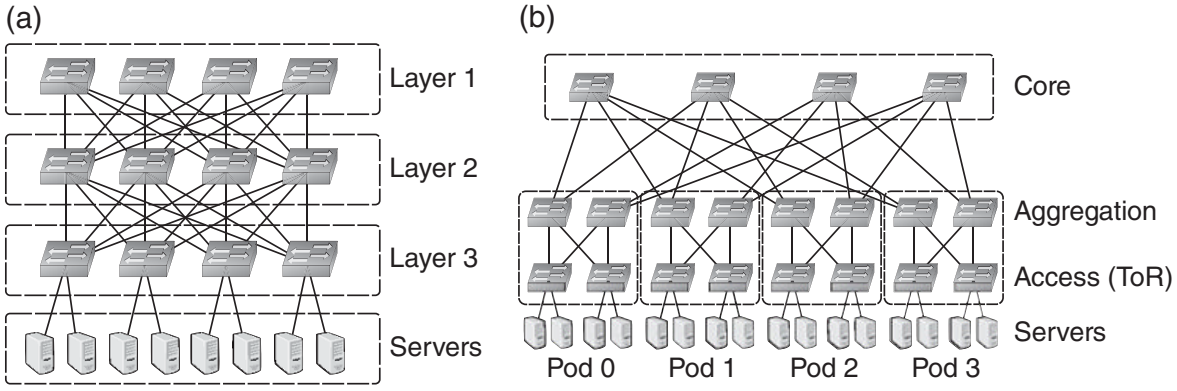

Figure 4.2. Clos-based topologies. (a) VL2 and (b) Fat-tree.

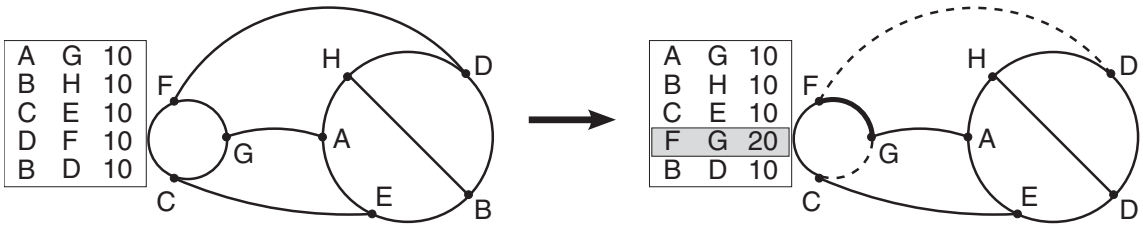

Figure 4.3. OSA adapts according to demands (adapted from Ref. [17]).

thousands of servers in a container, and latency-sensitive flows may be affected by link reconfiguration delays.

4.2.3 Hybrid Switch/Server Topologies

These architectures shift complexity from network devices to servers, i.e., servers perform routing, while low-end mini-switches interconnect a fixed number of hosts. They can also provide higher fault-tolerance, richer connectivity and improve innovation, because hosts are easier to customize than commodity switches. Two example topologies are DCell [5] and BCube [18], which can arguably scale up to millions of servers.

DCell [5] is a recursively built structure that forms a fully connected graph using only commodity switches (as opposed to high-end switches of traditional DCNs). DCell aims to scale out to millions of servers with few recursion levels (it can hold 3.2 million servers with only four levels and six hosts per cell). A DCell network is built as follows. A level-0 DCell (DCell_0) comprises servers connected to a n-port commodity switch. DCell_1 is formed with n + 1 DCell_0; each DCell_0 is connected to all other DCell_0 with one bidirectional link. In general, a level-k DCell is constructed with n + 1 DCell_{k-1} in the same manner as DCell_1. Figure 4.4a shows an example of a two-level DCell topology. In this example, a commodity switch is connected with four servers ($n = 4$) and, therefore, a DCell_1 is constructed with 5 DCell_0. The set of DCell_0 is interconnected in the following way: each server is represented by the tuple (a_1, a_0), where a_1 and a_0 are level 1 and 0 identifiers, respectively; and a link is created between servers identified by the tuples $(i, j-1)$ and (j, i), for every i and every $j > i$.

Similarly to DCell, BCube [18] is a recursively built structure that is easy to design and upgrade. Additionally, BCube provides low latency and graceful degradation of bandwidth upon link and switch failure. In this structure, clusters (a set of servers interconnected by a switch) are interconnected by commodity switches in a hypercube-based topology. More specifically, BCube is constructed as follows: BCube_0 (level-0 BCube) consists of n servers connected by a n-port switch; BCube_1 is constructed from n BCube_0 and n n-port switches; and BCube_k is constructed from n BCube_{k-1} and n^k n-port switches. Each server is represented by the tuple (x_1, x_2), where x_1 is the cluster number and x_2 is the server number inside the cluster. Each switch, in turn, is represented by a tuple (y_1, y_2), where y_1 is the level number and y_2 is the switch number inside the level. Links are created by connecting the level-k port of the i-th server in the j-th BCube_{k-1} to the j-th port of the i-th level-k switch. An example of two-level BCube with $n = 4$ (4-port switches) is shown in Figure 4.4b.

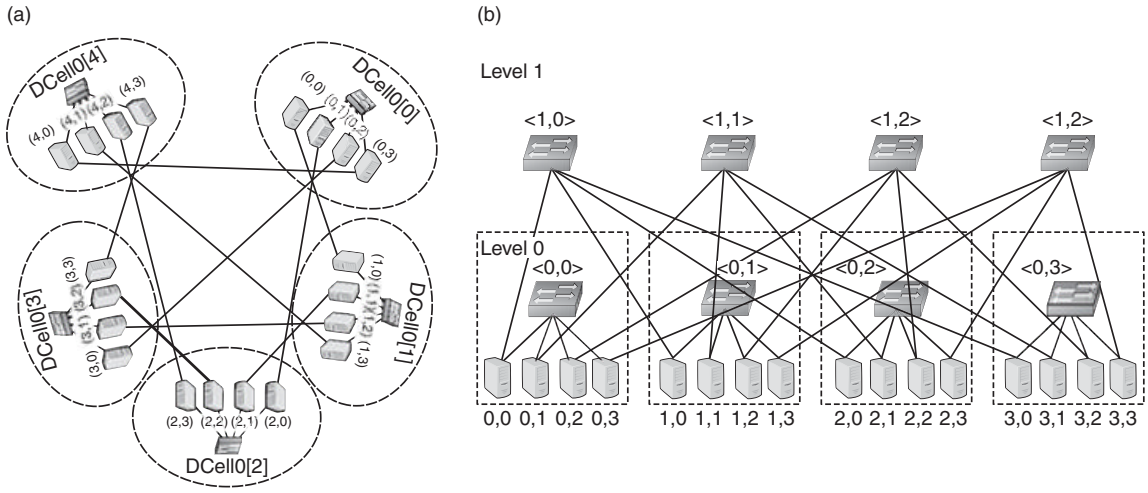

Figure 4.4. Hybrid switch/server topologies. (a) Two-level DCell and (b) two-level BCube.

Despite the benefits, DCell and BCube require a high number of NIC ports at end-hosts — causing some overhead at servers — and increase wiring costs. In particular, DCell results in non-uniform multiple paths between hosts, and level-0 links are typically more utilized than other links (creating bottlenecks). BCube, in turn, provides uniform multiple paths, but uses more switches and links than DCell [18].

4.2.4 Server-Only Topology

In this kind of topology, the network comprises only servers that perform all network functions. An example of architecture is CamCube [19], which is inspired in Content Addressable Network (CAN) [20] overlays and uses a 3D torus (k-ary 3-cube) topology with k servers along each axis. Each server is connected directly to 6 other servers, and the edge servers are wrapped. Figure 4.5 shows a 3-ary CamCube topology, resulting in 27 servers. The three most positive aspects of CamCube are (1) providing robust fault-tolerance guarantees (unlikely to partition even with 50% of server or link failures); (2) improving innovation with key-based server-to-server routing (content is hashed to a location in space defined by a server); and (3) allowing each application to define specific routing techniques. However, it does not hide topology from applications, has higher network diameter $O(\sqrt[3]{N})$ (increasing latency and traffic in the network) and hinders network expansion.

4.2.5 Summary of Topologies

Table 4.1 summarizes the benefits and limitations of these topologies by taking four properties into account: scalability, resiliency, agility and cost. The typical DCN topology has limited scalability (even though it can support hundreds of thousands of servers), as COTS switches have restricted memory size and need to maintain an entry in their Forwarding Information Base (FIB) for each VM. Furthermore, it presents low resiliency, since it provides only 1+1 redundancy, and its oversubscribed nature hinders agility.

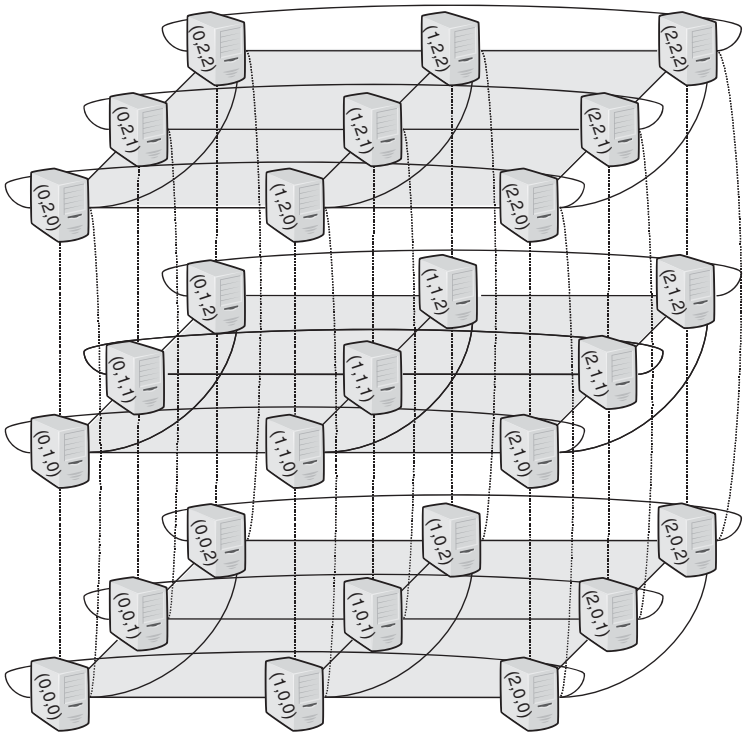

Figure 4.5. Example of 3-ary CamCube topology (adapted from Ref. [21]).

TABLE 4.1. Comparison among datacenter network topologies

Proposal	Properties			
	Scalability	Resiliency	Agility	Cost
Typical DCN	Low	Low	No	Low
Fat-Tree	High	Average	Yes	Average
VL2	High	High	Yes	High
OSA	Low	High	No	High
DCell	Huge	High	No	High
BCube	Huge	High	Yes	High
CamCube	Low	High	No	Average

Despite the drawbacks, it can be implemented with only commodity switches, resulting in lower costs.

Fat-Tree and VL2 are both instances of a Clos topology with high scalability and full bisection bandwidth (guaranteed agility). Fat-Tree achieves average resiliency, as ToR switches are connected only to a subset of aggregation devices, and has average overall costs (mostly because of increased wiring). VL2 scales through packet encapsulation, maintaining forwarding state only for switches in the network, achieves high resiliency by providing multiple shortest paths and by relying on a distributed lookup entity for handling address queries. As a downside, its deployment has increased costs (due to wiring, significant amount of exclusive resources for running the lookup system and the need of switch support for IP-in-IP encapsulation).

OSA was designed taking flexibility into account in order to improve resiliency (i.e., by using runtime reconfigurable optical devices to dynamically change physical topology and one-hop link capacities). However, it has low scalability (up to a few thousands of servers), no agility (as dynamically changing link capacities may result in congested links) and higher costs (devices should support optical reconfiguration).

DCell and BCube aim at scaling to millions of servers while ensuring high resiliency (rich connectivities between end-hosts). In contrast to BCube, DCell does not provide agility, as the set set of non-uniform multiple paths may be bottlenecked by links at level-0. Finally, their deployment costs may be significant, since they require a lot of wiring and more powerful servers in order to efficiently perform routing.

CamCube, in turn, is unlikely to partition even with 50% of server or link failures, thus achieving high resiliency. Its drawback, however, is related to scalability and agility; both properties can be hindered because of high network diameter, which indicates that, on average, more resources are needed for communication between VMs hosted by different servers. CamCube also has average deployment costs, mainly due to wiring and the need of powerful servers (to perform network functions).

As we can see, there is no perfect topology, since each proposal focus on specific aspects. Ultimately, providers are cost-driven: they choose the topology with the lowest costs, even if it cannot achieve all properties desired for a datacenter network running heterogenous applications from many tenants.

4.3 NETWORK EXPANSION

A key challenge concerning datacenter networks is dealing with the harmful effects that their ever-growing demand causes on scalability and performance. Because current DCN topologies are restricted to 1+1 redundancy and suffer from oversubscription, they can become underprovisioned quite fast. The lack of available bandwidth, in turn, may cause resource fragmentation (since it limits VM placement) [11] and reduce server utilization (as computations often depend on the data received from the network) [2]. In consequence, the DCN can loose its ability to accommodate more tenants (or offer elasticity to the current ones); even worse, applications using the network may start performing poorly, as they often rely on strict network guarantees[2].

These fundamental shortcomings have stimulated the development of novel DCN architectures (seen in Section 4.2) that provide large amounts of (or full) bisection bandwidth for up to millions of servers. Despite achieving high bisection bandwidth, their deployment is hindered by the assumption of homogeneous sets of switches (with the same number of ports). For example, consider a Fat-Tree topology, where the entire structure is defined by the number of ports in switches. These homogeneous switches limit the structure in two ways: full bisection bandwidth can only be achieved with

[2]For example, user-facing applications, such as Web services, require low-latency for communication with users, while inward computation (e.g., Map-Reduce) requires reliability and bisection bandwidth in the intra-cloud network.

specific numbers of servers (e.g., 8,192 and 27,648) and incremental upgrade may require replacing every switch in the network [13].

In fact, most physical datacenter designs are unique; hence, expansions and upgrades must be custom-designed and network performance (including bisection bandwidth, end-to-end latency and reliability) must be maximized while minimizing provider costs [11, 12]. Furthermore, organizations need to be able to incrementally expand their networks to meet the growing demands of tenants [13]. These facts have motivated recent studies [7, 11–13] to develop techniques to expand current DCNs to boost bisection bandwidth and reliability with heterogeneous sets of devices (i.e., without replacing every router and switch in the network). They are discussed next.

4.3.1 Legup

Focused on tree-like networks, Legup [12] is a system that aims at maximizing network performance at the design of network upgrades and expansions. It utilizes a linear model that combines three metrics (agility, reliability and flexibility), while being subject to the cloud provider's budget and physical constraints. In an attempt to reduce costs, the authors of Legup develop the *Theory of Heterogeneous Clos Networks* to allow modern and legacy equipment to coexist in the network. Figure 4.6 depicts an overview of the system. Legup assumes an existing set of racks and, therefore, only needs to determine aggregation and core levels of the network (more precisely, the set of devices, how they interconnect, and how they connect to ToR switches). It employs a branch and bound optimization algorithm to explore the solution space only for aggregation switches, as core switches in a heterogeneous Clos network are restricted by aggregation ones. Given a set of aggregation switches in each step of the algorithm, Legup performs three actions. First, it computes the minimum cost for mapping aggregation switches to racks. Second, it finds the minimum cost distribution of core switches to connect to the set of aggregation switches. Third, the candidate solution is bounded to check its optimality and feasibility (by verifying if any constraint is violated, including provider's budget and physical restrictions).

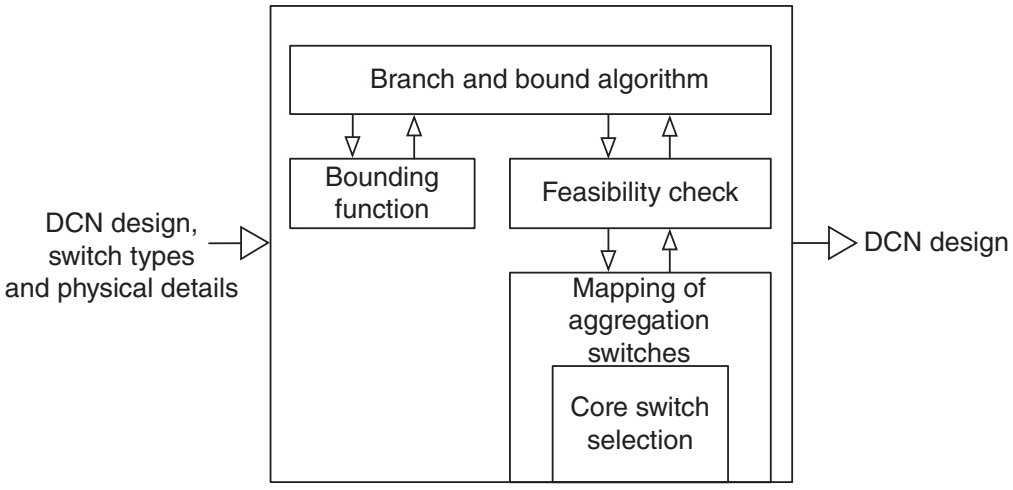

Figure 4.6. Legup's overview (adapted from Ref. [12]).

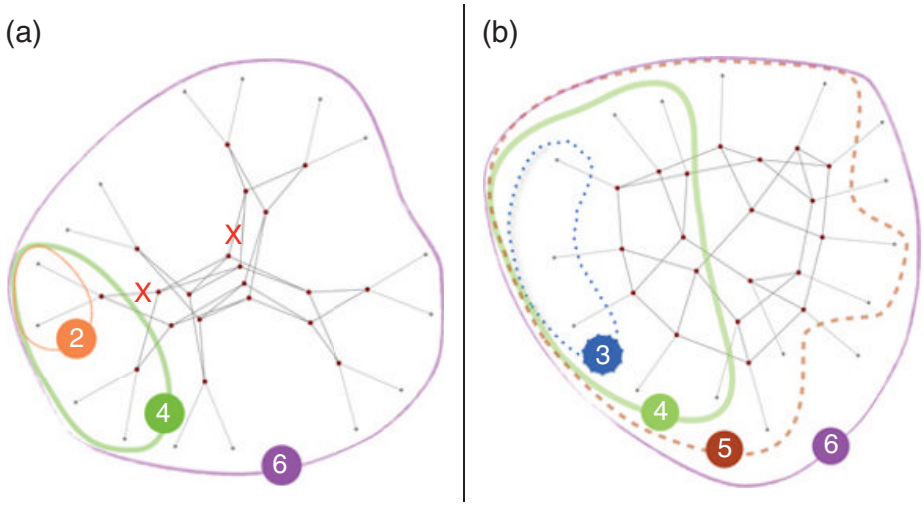

Figure 4.7. Comparison between (a) Fat-Tree and (b) Jellyfish with identical equipment (adapted from Ref. [13]).

4.3.2 Rewire

Recent advancements in routing protocols may allow DCNs to shift from a rigid tree to a generic structure [11, 22–25]. Based on this observation, Rewire [11] is a framework that performs DCN expansion on arbitrary topologies. It has the goal of maximizing network performance (i.e., finding maximum bisection bandwidth and minimum end-to-end latency), while minimizing costs and satisfying user-defined constraints. In particular, Rewire adopts a different definition of latency: while other studies model it by the worst-case hop-count in the network, Rewire also considers the speed of links and the processing time at switches (because unoptimized switches can add an order of magnitude more processing delay). Rewire uses simulated annealing (SA) [26] to search through candidate solutions and implements an approximation algorithm to efficiently compute their bisection bandwidth. The simulated annealing, however, does not take the addition of switches into account; it only optimizes the network for a given set of switches. Moreover, the process assumes uniform queuing delays for all switch ports, which is necessary because Rewire does not possess knowledge of network load.

4.3.3 Jellyfish

End-to-end throughput of a network is quantitatively proved to depend on two factors: (1) the capacity of the network and (2) the average path length (i.e., throughput is inversely proportional to the capacity consumed to deliver each byte) [13]. Furthermore, as noted earlier, rigid DCN structures hinder incremental expansion. Consequently, a degree-bounded[3] random graph topology among ToR switches, called Jellyfish [13], is introduced, with the goal of providing high bandwidth and flexibility. It supports device heterogeneity, different degrees of oversubscription and easy incremental expansion (by naturally allowing the addition of heterogeneous devices). Figure 4.7 shows a comparison

[3]Degree-bounded, in this context, means that the number of connections per node is limited by the number of ports in switches.

of Fat-Tree and Jellyfish with identical equipment and same diameter (i.e., 6). Each ring in the figure contains servers reachable within the number of hops in the labels. We see that Jellyfish can reach more servers in fewer hops, because some links are not useful from a path-length perspective in a Fat-Tree (e.g., links marked with "x"). Despite its benefits, Jellyfish's random design brings up some challenges, such as routing and the physical layout. Routing, in particular, is a critical feature needed, because it allows the use of the topology's high capacity. However, results show that the commonly used ECMP does not utilize the entire capacity of Jellyfish, and the authors propose the use of k-shortest paths and MultiPath TCP [25] to improve throughput and fairness.

4.3.4 Random Graph-Based Topologies

Singla et al. [7] analyze the throughput achieved by random graphs for topologies with both homogeneous and heterogeneous switches, while taking optimization into account. They obtain the following results: random graphs achieve throughput close to the optimal upper-bound under uniform traffic patterns for homogeneous switches, and heterogeneous networks with distinct connectivity arrangements can provide nearly identical high throughput. Then, the acquired knowledge is used as a building block for designing large-scale random networks with heterogeneous switches. In particular, they utilize the VL2 deployed in Microsoft's datacenters as a case study, showing that its throughput can be significantly improved (up to 43%) by only rewiring the same devices.

4.4 TRAFFIC

Proposals of topologies for datacenter networks presented in Sections 4.2 and 4.3 share a common goal: provide high bisection bandwidth for tenants and their applications. It is intuitive that a higher bisection bandwidth will benefit tenants, since the communication between VMs will be less prone to interference. Nonetheless, it is unclear how strong is the impact of the bisection bandwidth. This section addresses this question by surveying several recent measurement studies of DCNs. Then, it reviews proposals for dealing with related limitations. More specifically, it discusses traffic patterns—highlighting their properties and implications for both providers and tenants—and shows how literature is using such information to help designing and managing DCNs.

Traffic can be divided in two broad categories: north/south and east/west communication. North/south traffic (also known as extra-cloud) corresponds to the communication between a source and a destination host where one of the ends is located outside the cloud platform. By contrast, east/west traffic (also known as intra-cloud) is the communication in which both ends are located inside the cloud. These types of traffic usually depend on the kind and mix of applications: user-facing applications (e.g., web services) typically exchange data with users and, thus, generate north/south communication, while inward computation (i.e., MapReduce) requires coordination among its VMs, generating east/west communication. Studies [27] indicate that north/south and east-west traffic correspond to around 25% and 75% of traffic volume, respectively. They also point that

both are increasing in absolute terms, but east/west is growing on a larger scale [27]. Towards understanding traffic characteristics and how it influences the proposal of novel mechanisms, we first discuss traffic properties defined by measurement studies in the literature [9, 28–30] and, then, examine traffic management and its most relevant proposals for large-scale cloud datacenters.

4.4.1 Properties

Traffic in the cloud network is characterized by flows; each flow is identified by sequences of packets from a source to a destination node (i.e., a flow is defined by a set packet header fields, such as source and destination addresses and ports and transport protocol). Typically, a bimodal flow classification scheme is employed, using elephant and mice classes. Elephant flows comprise a large number of packets injected in the network over a short amount of time, are usually long-lived and exhibit bursty behavior. In comparison, mice flows have a small number of packets and are short-lived [3]. Several measurement studies [9, 28–31] were conducted to characterize network traffic and its flows. We summarize their findings as follows:

- *Traffic asymmetry*. Requests from users to cloud services are abundant, but small in most occasions. Cloud services, however, process these requests and typically send responses that are comparatively larger.
- *Nature of traffic*. Network traffic is highly volatile and bursty, with links running close to their capacity at several times during a day. Traffic demands change quickly, with some transient spikes and other longer ones (possibly requiring more than half the full-duplex bisection bandwidth) [32]. Moreover, traffic is unpredictable at long time scales (e.g., 100 seconds or more). However, it can be predictable on shorter timescales (at 1 or 2 seconds). Despite the predictability over small timescales, it is difficult for traditional schemes, such as statistical multiplexing, to make a reliable estimate of bandwidth demands for VMs [33].
- *General traffic location and exchange*. Most traffic generated by servers (on average 80%) stays within racks. Server pairs from the same rack and from different racks exchange data with a probability of only 11% and 0.5%, respectively. Probabilities for intra- and extra-rack communication are as follows: servers either talk with fewer than 25% or to almost all servers of the same rack; and servers communicate with less than 10% or do not communicate with servers located outside its rack.
- *Intra- and inter-application communication*. Most volume of traffic (55%) represents data exchange between different applications. However, the communication matrix between them is sparse; only 2% of application pairs exchange data, with the top 5% of pairs accounting for 99% of inter-application traffic volume. Consequently, communicating applications form several highly connected components, with few applications connected to hundreds of other applications in star-like topologies. In comparison, intra-application communication represents 45% of the total traffic, with 18% of applications generating 99% of this traffic volume.

- *Flow size, duration, and number*. Mice flows represent around 99% of the total number of flows in the network. They usually have less than 10 kilobytes and last only a few hundreds of milliseconds. Elephant flows, in turn, represent only 1% of the number of flows, but account for more than half of the total traffic volume. They may have tens of megabytes and last for several seconds. With respect to flow duration, flows of up to 10 seconds represent 80% of flows, while flows of 200 seconds are less than 0.1% (and contribute to less than 20% of the total traffic volume). Further, flows of 25 seconds or less account for more than 50% of bytes. Finally, it has been estimated that a typical rack has around 10,000 active flows per second, which means that a network comprising 100,000 servers can have over 25,000,000 active flows.
- *Flow arrival patterns*. Arrival patterns can be characterized by heavy-tailed distributions with a positive skew. They best fit a log-normal curve having ON and OFF periods (at both 15 and 100 milliseconds granularities). In particular, inter arrival times at both servers and ToR switches have periodic modes spaced apart by approximately 15 milliseconds, and the tail of these distributions is long (servers may experience flows spaced apart by 10 seconds).
- *Link utilization*. Utilization is, on average, low in all layers but the core; in fact, in the core, a subset of links (up to 25% of all core links) often experience high utilization. In general, link utilization varies according to temporal patterns (time of day, day of week and month of year), but variations can be an order of magnitude higher at core links than at aggregation and access links. Due to these variations and the bursty nature of traffic, highly utilized links can happen quite often; 86% and 15% of links may experience congestion lasting at least 10 and 100 seconds, respectively, while longer periods of congestion tend to be localized to a small set of links.
- *Hot spots*. They are usually located at core links and can appear quite frequently, but the number of hot spots never exceeds 25% of core links.
- *Packet losses*. Losses occur frequently even at underutilized links. Given the bursty nature of traffic, an underutilized network (e.g., with mean load of 10%) can experience lots of packet drops. Measurement studies found that packet losses occur usually at links with low average utilization (but with traffic bursts that go beyond 100% of link capacity); more specifically, such behavior happens at links of the aggregation layer and not at links of the access and core layers. Ideally, topologies with full bisection bandwidth (i.e., a Fat-Tree) should experience no loss, but the employed routing mechanisms cannot utilize the full capacity provided by the set of multiple paths and, consequently, there is some packet loss in such networks as well [28].

4.4.2 Traffic Management

Other set of papers [34–37] demonstrate that available bandwidth for VMs inside the datacenter can vary by a factor of five or more in the worst-case scenario. Such variability results in poor and unpredictable network performance and reduced overall application

performance [1, 38, 39], since VMs usually depend on the data received from the network to execute the subsequent computation.

The lack of bandwidth guarantees is related to two main factors. First, the canonical cloud topology is typically oversubscribed, with more bandwidth available in leaf nodes than in the core. When periods of traffic bursts happen, the lack of bandwidth up the tree (i.e., at aggregation and core layers) results in contention and, therefore, packet discards at congested links (leading to subsequent retransmissions). Since the duration of the timeout period is typically one or two orders of magnitude more than the round-trip time, latency is increased, becoming a significant source of performance variability [3]. Second, TCP congestion control does not provide robust isolation among flows. Consequently, elephant flows can cause contention in congested links shared with mice flows, leading to discarded packets from the smaller flows [2].

Recent proposals address this issue either by employing proportional sharing or by providing bandwidth guarantees. Most of them use the hose model [40] for network virtualization and take advantage of rate-limiting at hypervisors [41], VM placement [42] or virtual network embedding [43] in order to increase their robustness.

Proportional sharing. Seawall [2] and NetShare [44] allocate bandwidth at flow-level based on weights assigned to entities (i.e., VMs or services running inside these VMs) that generate traffic in the network. While both assign weights based on administrator specified policies, NetShare also supports automatic weight assignment. Both schemes are work-conserving (i.e., available bandwidth can be used by any flow that needs more bandwidth), provide max–min fair sharing and achieve high utilization through statistical multiplexing. However, as bandwidth allocation is performed per flow, such methods may introduce substantial management overhead in large datacenter networks (with over 10,000 flows per rack per second [9]). FairCloud [45] takes a different approach and proposes three allocation policies to explore the trade-off among network proportionality, minimum guarantees and high utilization. Unlike Seawall and NetShare, FairCloud does not allocate bandwidth along congested links at flow-level, but in proportion to the number of VMs of each tenant. Despite the benefits, FairCloud requires customized hardware in switches and is designed specifically for tree-like topologies.

Bandwidth guarantees. SecondNet [46], Gatekeeper [47], Oktopus [1], Proteus [48], and Hadrian [49] provide minimum bandwidth guarantees by isolating applications in virtual networks. In particular, SecondNet is a virtualization architecture that distributes the virtual-to-physical mapping, routing and bandwidth reservation state in server hypervisors. Gatekeeper configures each VM virtual NIC with both minimum and maximum bandwidth rates, which allows the network to be shared in a work-conserving manner. Oktopus maps tenants' virtual network requests (with or without oversubscription) onto the physical infrastructure and enforces these mappings in hypervisors. Proteus is built based on the observation that allocating the peak bandwidth requirements for applications leads to underutilization of resources. Hence, it quantifies the temporal bandwidth demands of applications and allocates each one of them in a different virtual network. Hadrian extends previous schemes by also taking inter-tenant communication into account and allocating applications according to a hierarchical hose model (i.e., per VM minimum bandwidth for intra-application communication and per tenant minimum guarantees for inter-tenant traffic). By contrast, a group of related proposals attempt to

provide some level bandwidth sharing among applications of distinct tenants [50–52]. The approach introduced by Marcon et al. [51] groups applications in virtual networks, taking mutually trusting relationships between tenants into account when allocating each application. It provides work-conserving network sharing, but assumes that trust relationships are determined in advance. ElasticSwitch [52] assumes there exists an allocation method in the cloud platform and focuses on providing minimum bandwidth guarantees with a work-conserving sharing mechanism (when there is spare capacity in the network). Nevertheless, it requires two extra management layers for defining the amount of bandwidth for each flow, which may add overhead. Finally, EyeQ [50] leverages high bisection bandwidth of DCNs to support minimum and maximum bandwidth rates for VMs. Therefore, it provides work-conserving sharing, but depends on the core of the network to be congestion-free. None of these approaches can be readily deployed, as they demand modifications in hypervisor source code.

4.5 ROUTING

Datacenter networks often require specially tailored routing protocols, with different requirements from traditional enterprise networks. While the latter presents only a handful of paths between hosts and predictable communication patterns, DCNs require multiple paths to achieve horizontal scaling of hosts with unpredictable traffic matrices [4, 6]. In fact, datacenter topologies (i.e., the ones discussed in Section 4.2) typically present path diversity, in which multiple paths exist between servers (hosts) in the network. Furthermore, many cloud applications (ranging from Web search to MapReduce) require substantial (possibly full bisection) bandwidth [53]. Thus, routing protocols must enable the network to deliver high bandwidth by using all possible paths in the structure. We organize the discussion according to the layer involved, starting with the network layer.

4.5.1 Layer 3

To take advantage of the multiple paths available between a source and its destination, providers usually employ two techniques: ECMP [54] and VLB [6, 55, 56]. Both strategies use distinct paths for different flows. ECMP attempts to load balance traffic in the network and utilize all paths which have the same cost (calculated by the routing protocol) by uniformly spreading traffic among them using flow hashing. VLB randomly selects an intermediate router (occasionally, a L3 switch) to forward the incoming flow to its destination.

Recent studies in the literature [46, 53, 57, 58] propose other routing techniques for DCNs. As a matter of fact, the static flow-to-path mapping performed by ECMP does not take flow size and network utilization into account [59]. This may result in saturating commodity switch L3 buffers and degrading overall network performance [53]. Therefore, a system called Hedera [53] is introduced to allow dynamic flow scheduling for general multi-rooted trees with extensive path diversity. Hedera is designed to maximize

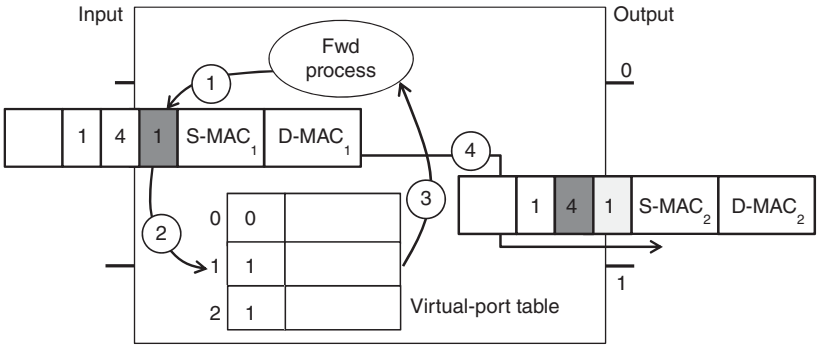

Figure 4.8. PSSR overview (adapted from Ref. [46]).

network utilization with low scheduling overhead of active flows. In general, the system performs the following steps: (1) detects large flows at ToR switches; (2) estimates network demands of these large flows (with a novel algorithm that considers bandwidth consumption according to a max–min fair resource allocation); (3) invokes a placement algorithm to compute paths for them; and (4) installs the set of new paths on switches.

Hedera uses a central OpenFlow controller[4] [60] with a global view of the network to query devices, obtain flow statistics and install new paths on devices after computing their routes. With information collected from switches, Hedera treats the flow-to-path mapping as an optimization problem and uses a simulated annealing metaheuristic to efficiently look for feasible solutions close to the optimal one in the search space. SA reduces the search space by allowing only a single core switch to be used for each destination. Overall, the system delivers close to optimal performance and up to four times more bandwidth than ECMP.

Port-switching based source routing (PSSR) [46] is proposed for the SecondNet architecture with arbitrary topologies and commodity switches. PSSR uses source routing, which requires that every node in the network knows the complete path to reach a destination. It takes advantage of the fact that a datacenter is administered by a single entity (i.e., the intra-cloud topology is known in advance) and represents a path as a sequence of output ports in switches, which is stored in the packet header. More specifically, the hypervisor of the source VM inserts the routing path in the packet header, commodity switches perform the routing process with PSSR and the destination hypervisor removes PSSR information from the packet header and delivers the packet to the destination VM. PSSR also introduces the use of virtual ports, because servers may have multiple neighbors via a single physical port (e.g., in DCell and BCube topologies). The process performed by a switch is shown in Figure 4.8. Switches read the pointer field in the packet header to get the exact next output port number (step 1), verify the next port number in the lookup virtual-port table (step 2), get the physical port number (step 3) and, in step 4, update the pointer field and forward the packet. This routing method introduces some overhead (since routing information must be included in the packet header), but, according to the authors, can be easily implemented on commodity switches using Multi-Protocol Label Switching (MPLS) [61].

[4]We will not focus our discussion in OpenFlow in this chapter. It is discussed in Chapter 6.

Bounded Congestion Multicast Scheduling (BCMS) [57], introduced to efficiently route flows in Fat-trees under the hose traffic model, aims at achieving bounded congestion and high network utilization. By using multicast, it can reduce traffic, thus minimizing performance interference and increasing application throughput [62]. BCMS is an online multicast scheduling algorithm that leverages OpenFlow to (1) collect bandwidth demands of incoming flows; (2) monitor network load; (3) compute routing paths for each flow; and (4) configure switches (i.e., installing appropriate rules to route flows). The algorithm has three main steps, as follows. First, it checks the conditions of uplinks out of source ToR switches (as flows are initially routed towards core switches). Second, it carefully selects a subset of core switches in order to avoid congestion. Third, it further improves traffic load balance by allowing ToR switches to connect to core switches with most residual bandwidth. Despite its advantages, BCMS relies on a centralized controller, which may not scale to large datacenters under highly dynamic traffic patterns such as the cloud.

Like BCMS, Code-Oriented eXplicit multicast (COXcast) [58] also focuses on routing application flows through the use of multicasting techniques (as a means of improving network resource sharing and reducing traffic). COXcast uses source routing, so all information regarding destinations are added to the packet header. More specifically, the forwarding information is encoded into an identifier in the packet header and, at each network device, is resolved into an output port bitmap by a node-specific key. COXcast can support a large number of multicast groups, but it adds some overhead to packets (since all information regarding routing must be stored in the packet).

4.5.2 Layer 2

In the Spanning Tree Protocol (STP) [63], all switches agree on a subset of links to be used among them, which forms a spanning tree and ensures a loop-free network. Despite being typically employed in Ethernet networks, it does not scale, since it cannot use the high-capacity provided by topologies with rich connectivities (i.e., Fat-Trees [24]), limiting application network performance [64]. Therefore, only a single path is used between hosts, creating bottlenecks and reducing overall network utilization.

STP's shortcomings are addressed by other protocols, including Multiple Spanning Tree Protocol (MSTP) [65], Transparent Interconnect of Lots of Links (TRILL) [22] and Link Aggregation Control Protocol (LACP) [66]. MSTP was proposed in an attempt to use the path diversity available in DCNs more efficiently. It is an extension of STP to allow switches to create various spanning trees over a single topology. Therefore, different Virtual LANs (VLANs) [67] can utilize different spanning trees, enabling the use of more links in the network than with a single spanning tree. Despite its objective, implementations only allow up to 16 different spanning trees, which may not be sufficient to fully utilize the high-capacity available in DCNs [68].

TRILL is a link-state routing protocol implemented on top of layer-2 technologies, but bellow layer-3, and is designed specifically to address limitations of STP. It discovers and calculates shortest paths between TRILL devices (called routing bridges or, in short, RBridges), which enables shortest path multihop routing in order to use all available paths

in networks with rich connectivities. RBridges run Intermediate System to Intermediate System (IS-IS) routing protocol (RFC 1195) and handle frames in the following manner: the first RBridge (ingress node) encapsulates the incoming frame with a TRILL header (outer MAC header) that specifies the last TRILL node as the destination (egress node), which will decapsulate the frame.

Link Aggregation Control Protocol (LACP) is another layer-2 protocol used in DCNs. It transparently aggregates multiple physical links into one logical link known as Link Aggregation Group (LAG). LAGs only handle outgoing flows; they have no control over incoming traffic. They provide flow-level load balancing among links in the group by hashing packet header fields. LACP can dynamically add or remove links in LAGs, but requires that both ends of a link run the protocol.

There are also some recent studies that propose novel strategies for routing frames in DCNs, namely Smart Path Assignment in Networks (SPAIN) [24] and Portland [64]. SPAIN [24] focuses on providing efficient multipath forwarding using COTS switches over arbitrary topologies. It has three components: (1) path computation; (2) path setup; and (3) path selection. The first two components run on a centralized controller with global network visibility. The controller first pre-computes a set of paths to exploit the rich connectivities in the DCN topology, in order to use all available capacity of the physical infrastructure and to support fast failover. After the path computation phase, the controller combines these multiple paths into a set of trees, with each tree belonging to a distinct VLAN. Then, these VLANs are installed on switches. The third component (path selection) runs at end-hosts for each new flow; it selects paths for flows with the goals of spreading load across the pre-computed routes (by the path setup component) and minimizing network bottlenecks. With this configuration, end-hosts can select different VLANs for communication (i.e., different flows between the same source and destination can use distinct VLANs for routing). To provide these functionalities, however, SPAIN requires some modification to end-hosts, adding an algorithm to choose among pre-installed paths for each flow.

PortLand [64] is designed and built based on the observation that Ethernet/IP protocols may have some inherent limitations when designing large-scale arbitrary topologies, such as limited support for VM migration, difficult management and inflexible communication. It is a layer-2 routing and forwarding protocol with plug-and-play support for multi-rooted Fat-Tree topologies. PortLand uses a logically centralized controller (called fabric manager) with global visibility and maintains soft state about network configuration. It assigns unique hierarchical Pseudo MAC (PMAC) addresses for each VM to provide efficient, provably loop-free frame forwarding; VMs, however, do not have the knowledge of their PMAC and believe they use their Actual MAC (AMAC). The mapping between PMAC and AMAC and the subsequent frame header rewriting is performed by edge (ToR) switches. PMACs are structured as *pod.position.port.vmid*, where each field respectively corresponds to the pod number of the edge switch, its position inside the pod, the port number in which the physical server is connected to and the identifier of the VM inside the server. With PMACs, PortLand transparently provides location-independent addresses for VMs and requires no modification in commodity switches. However, it has two main shortcomings (1) it requires a Fat-Tree topology (instead of the traditional multi-rooted oversubscribed tree) and (2) at least half of the

ToR switch ports should be connected to servers (which, in fact, is a limitation of Fat-Trees) [69].

4.6 ADDRESSING

Each server (or, more specifically, each VM) must be represented by a unique canonical address that enables the routing protocol to determine paths in the network. Cloud providers typically employ LAN technologies for addressing VMs in datacenters, which means there is a single address space to be sliced among tenants and their applications. Consequently, tenants have neither flexibility in designing their application layer-2 and layer-3 addresses nor network isolation from other applications.

Some isolation is achieved by the use of VLANs, usually one VLAN per tenant. However, VLANs are ill-suited for datacenters for four main reasons [51, 70–72]: (1) they do not provide flexibility for tenants to design their layer-2 and layer-3 address spaces; (2) they use the spanning tree protocol, which cannot utilize the high-capacity available in DCN topologies (as discussed in the previous section); (3) they have poor scalability, since no more than 4094 VLANs can be created, and this is insufficient for large datacenters; and *iv*) they do not provide location-independent addresses for tenants to design their own address spaces (independently of other tenants) and for performing seamless VM migration. Therefore, providers need to use other mechanisms to allow address space flexibility, isolation and location independence for tenants while multiplexing them in the same physical infrastructure. We structure the discussion in three main topics: emerging technologies, separation of name and locator and full address space virtualization.

4.6.1 Emerging Technologies

Some technologies employed in DCNs are: Virtual eXtensible Local Area Network (VXLAN) [73], Amazon Virtual Private Cloud (VPC) [74] and Microsoft Hyper-V [75]. VXLAN [73] is an Internet draft being developed to address scalability and multipath usage in DCNs when providing logical isolation among tenants. VXLAN works by creating overlay (virtual layer-2) networks on top of the actual layer-2 or on top of UDP/IP. In fact, using MAC-in-UDP encapsulation abstracts VM location (VMs can only view the virtual layer-2) and, therefore, enables a VXLAN network to be composed of nodes within distinct domains (DCNs), increasing flexibility for tenants using multi-datacenter cloud platforms. VXLAN adds a 24-bit segment ID field in the packet header (allowing up to 16 million different logical networks), uses ECMP to distribute load along multiple paths and requires Internet Group Management Protocol (IGMP) for forwarding frames to unknown destinations, or multicast and broadcast addresses. Despite the benefits, VXLAN header adds 50 bytes to the frame size, and multicast and network hardware may limit the usable number of overlay networks in some deployments.

Amazon VPC [74] provides full IP address space virtualization, allowing tenants to design layer-3 logical isolated virtual networks. However, it does not virtualize layer-2,

which does not allow tenants to send multicast and broadcast frames [71]. Microsoft Hyper-V [75] is a hypervisor-based system that provides virtual networks for tenants to design their own address spaces; Hyper-V enables IP overlapping in different virtual networks without using VLANs. Furthermore, Hyper-V switches are software-based layer-2 network switches with capabilities to connect VMs among themselves, with other virtual networks and with the physical network. Hyper-V, nonetheless, tends to consume more resources than other hypervisors with the same load [76].

4.6.2 Separation of Name and Locator

VL2 [6] and Crossroads [70] focus on providing location independence for VMs, so that providers can easily grow or shrink allocations and migrate VMs inside or across datacenters. VL2 [6] uses two types of addresses: location-specific addresses (LAs), which are the actual addresses in the network, used for routing; and application-specific addresses (AAs), permanent address assigned to VMs that remain the same even after migrations. VL2 uses a directory system to enforce isolation among applications (through access control policies) and to perform the mapping between names and locators; each server with an AA is associated with the LA from the ToR it is connected to. Figure 4.9 depicts how address translation in VL2 is performed: the source hypervisor encapsulates the AA address with the LA address of the destination ToR for each packet sent; packets are forwarded in the network through shortest paths calculated by the routing protocol, using both ECMP and VLB; when packets arrive at the destination ToR switch, LAs are removed (packets are decapsulated) and original packets are sent to the correct VMs using AAs. To provide location-independent addresses, VL2 requires that hypervisors run a shim layer (VL2 agent) and that switches support IP-over-IP.

Crossroads [70], in turn, is a network fabric developed to provide layer agnostic and seamless VM migration inside and across DCNs. It takes advantage of

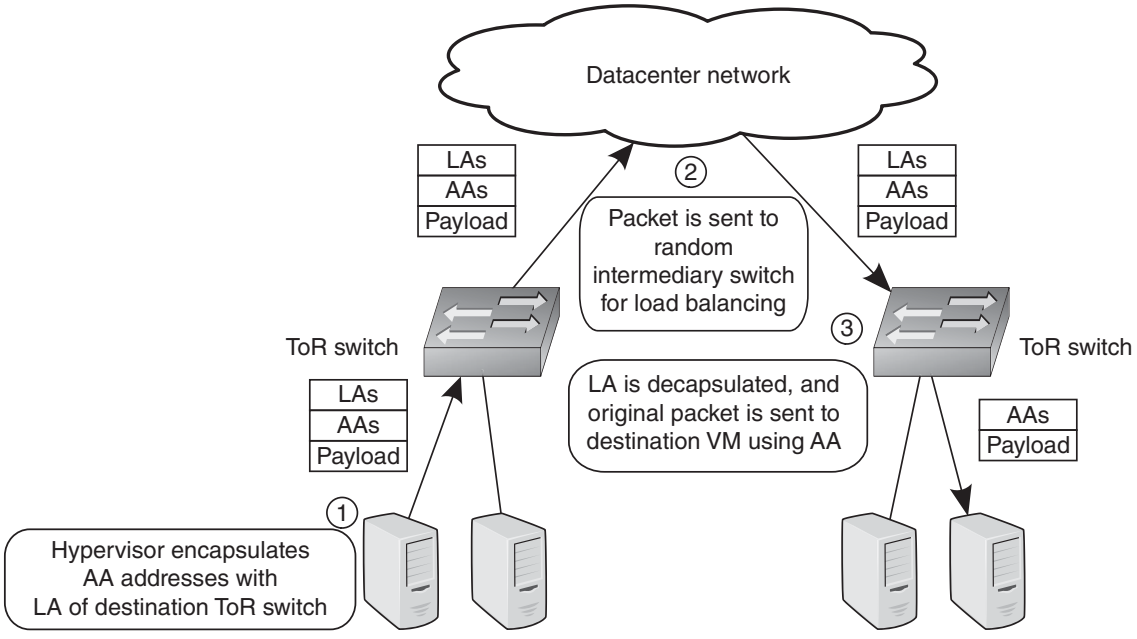

Figure 4.9. Architecture for address translation in VL2.

the Software-Defined Networking (SDN) paradigm [77] and extends an OpenFlow controller to allow VM location-independence without modifications to layer-2 and layer-3 network infrastructure. In Crossroads, each VM possess two addresses: a PMAC and a Pseudo IP (PIP), both with location and topological information embedded in them. The first one ensures that traffic originated from one datacenter and en route to a second datacenter (to which the VM was migrated) can be maintained at layer-2, while the second guarantees that all traffic destined to a migrated VM can be routed across layer-3 domains. Despite its benefits, Crossroads introduces some network overhead, as nodes must be identified by two more addresses (PMAC and PIP) in addition to the existing MAC and IP.

4.6.3 Full Address Space Virtualization

Cloud datacenters typically provide limited support for multi-tenancy, since tenants should be able to design their own address spaces (similar to a private environment) [71]. Consequently, a multi-tenant virtual datacenter architecture to enable specific-tailored layer-2 and layer-3 address spaces for tenants, called NetLord [71], is proposed. At hypervisors, NetLord runs an agent that performs Ethernet+IP (L2+L3) encapsulation over tenants' layer-2 frames and transfers them through the network using SPAIN [24] for multipathing, exploring features of both layers. More specifically, the process of encapsulating/decapsulating is shown in Figure 4.10 and occurs in three steps, as follows: (1) the agent at the source hypervisor creates L2 and L3 headers (with source IP being a tenant-assigned MAC address space identifier, illustrated as MAC_AS_ID) in order to direct frames through the L2 network to the correct edge switch; (2) the edge switch forwards the packet to the correct server based on the IP destination address in the virtualized layer-3 header; (3) the hypervisor at the destination server removes the virtual L2 and L3 headers and uses the IP destination address to deliver the original packet from the source VM to the correct VM. NetLord can be run on commodity switches and scale the network to hundreds of thousands of VMs. However, it requires an agent running on hypervisors (which may add some overhead) and support for IP forwarding on commodity edge (ToR) switches.

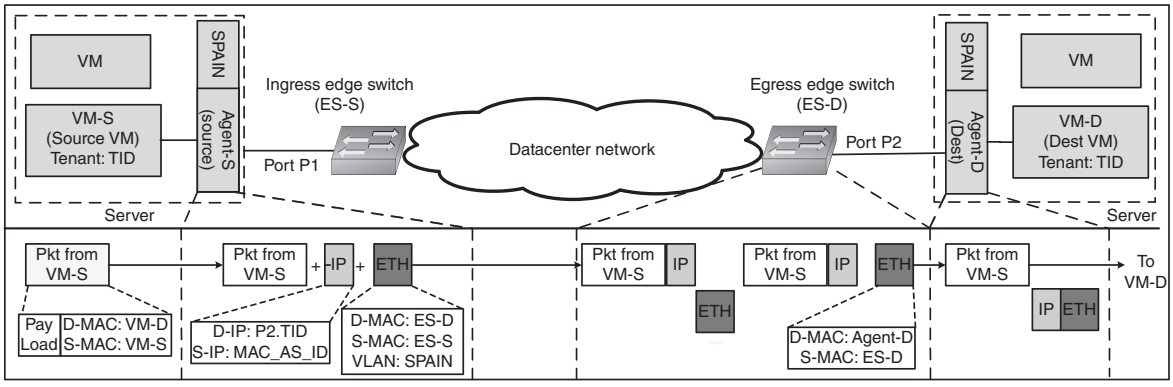

Figure 4.10. NetLord's encapsulation/decapsulation process (adapted from Ref. [71]).

4.7 RESEARCH CHALLENGES

In this section, we analyze and discuss open research challenges and future directions regarding datacenter networks. As previously mentioned, DCNs (i) present some distinct requirements from traditional networks (e.g., high scalability and resiliency); (ii) have significantly different (often more complex) traffic patterns; and (iii) may not be fully utilized, because of limitations in current deployed mechanisms and protocols (for instance, ECMP). Such aspects introduce some challenges, which are discussed next.

4.7.1 Heterogeneous and Optimal DCN Design

Presently, many Internet services and applications rely on large-scale datacenters to provide availability while scaling in and out according to incoming demands. This is essential in order to offer low response time for users, without incurring excessive costs for owners. Therefore, datacenter providers must build infrastructures to support large and dynamic numbers of applications and guarantee quality of service (QoS) for tenants. In this context, the network is an essential component of the whole infrastructure, as it represents a significant fraction of investment and contributes to future revenues by allowing efficient use of datacenter resources [15]. According to Zhang et al. [78], network requirements include (i) scalability, so that a large number of servers can be accommodated (while allowing incremental expansion); (ii) high server-to-server capacity, to enable intensive communication between any pair of servers (i.e., at full speed of their NICs); (iii) agility, so applications can use any available server when they need more resources (and not only servers located near their current VMs); (iv) uniform network utilization to avoid bottlenecks; and (v) fault tolerance to cope with server, switch and link failures. In fact, guaranteeing such requirements is a difficult challenge. Looking at these challenges from the providers viewpoint make them even more difficult to address and overcome, since reducing the cost of building and maintaining the network is seen as a key enabler for maximizing profits [15].

As discussed in Section 4.2, several topologies (e.g., Refs. [4–6, 17, 18]) have been proposed to achieve the desired requirements, with varying costs. Nonetheless, they (i) focus on homogeneous networks (all devices with the same capabilities); and (ii) do not provide theoretical foundations regarding optimality. Singla et al. [7], in turn, take an initial step towards addressing heterogeneity and optimality, as they (i) measure the upper-bound on network throughput for homogeneous topologies with uniform traffic patterns; and (ii) show an initial analysis of possible gains with heterogeneous networks. Despite this fact, a lot remains to be investigated in order to enable the development of more efficient, robust large-scale networks with heterogeneous sets of devices. In summary, very little is known about heterogeneous DCN design, even though current DCNs are typically composed of heterogenous equipment.

4.7.2 Efficient and Incremental Expansion

Providers need to be constantly expanding their datacenter infrastructures to accommodate ever-growing demands. For instance, Facebook has been expanding its datacenters

for some years [79–82]. This expansion is crucial for business, as the increase of demand may negatively impact scalability and performance (e.g., by creating bottlenecks in the network). When the whole infrastructure is upgraded, the network must be expanded accordingly, with a careful design plan, in order to allow efficient utilization of resources and to avoid fragmentation. To address this challenge, some proposals in the literature [7, 11–13] have been introduced to enlarge current DCNs without replacing legacy hardware. They aim at maximizing high bisection bandwidth and reliability. However, they often make strong assumptions (e.g., Legup [12] is designed for tree-like networks, and Jellyfish [13] requires new mechanisms for routing). Given the importance of datacenters nowadays (as home of hundreds of thousands of services and applications), the need for efficient and effective expansion of large-scale networks is a key challenge for improving provider profit, QoS offered to tenant applications and quality of experience (QoE) provided for users of these applications.

4.7.3 Network Sharing and Performance Guarantees

Datacenters host applications with diverse and complex traffic patterns and different performance requirements. Such applications range from user-facing ones (i.e., Web services and online gaming) that require low latency communication to inward computation (e.g., scientific computing) that need high network throughput. To gain better understanding of the environment, studies [1, 9, 30, 49, 83] conducted measurements and concluded that available bandwidth for VMs inside the cloud platform can vary by a factor of five or more during a predefined period of time. They demonstrate that such variability ends up impacting overall application execution time (resulting in poor and unpredictable performance). Several strategies (including Refs. [2, 47, 48, 52, 84]) have been proposed to address this issue. Nonetheless, they have one or more of the following shortcomings: (i) require complex mechanisms, which, in practice, cannot be deployed; (ii) focus on network sharing among VMs (or applications) in a homogeneous infrastructure (which simplifies the problem [85]); (iii) perform static bandwidth reservations (resulting in underutilization of resources); or (iv) provide proportional sharing (no strict guarantees). In fact, there is an inherent trade-off between providing strict guarantees (desired by tenants) and enabling work-conserving sharing in the network (desired by providers to improve utilization), which may be exacerbated in a heterogenous network. We believe this challenge requires further investigation, since such high-performance networks ideally need simple and efficient mechanisms to allow fair bandwidth sharing among running applications in a heterogeneous environment.

4.7.4 Address Flexibility for Tenants

While network performance guarantees require quantitative performance isolation, address flexibility needs qualitative isolation [71]. Cloud DCNs, however, typically provide limited support for multi-tenancy, as they have a single address space divided among applications (according to their needs and number of VMs). Thereby, tenants have no flexibility in choosing layer-2 and layer-3 addresses for applications. Note that,

ideally, tenants should be able to design their own address spaces (i.e., they should have similar flexibility to a private environment), since already developed applications may necessitate a specific set of addresses to correctly operate without source code modification. Some proposals in the literature [6, 70, 71] seek to address this challenge either by identifying end-hosts with two addresses or by fully virtualizing layer-2 and layer-3. Despite adding flexibility for tenants, they introduce some overhead (e.g., hypervisors need a shim layer to manage addresses, or switches must support IP-over-IP) and require resources specifically used for address translation (in the case of VL2). This is an important open challenge, as the lack of address flexibility may hinder the migration of applications to the cloud platform.

4.7.5 Mechanisms for Load Balancing Across Multiple Paths

DCNs usually present path diversity (i.e., multiple paths between servers) to achieve horizontal scaling for unpredictable traffic matrices (generated from a large number of heterogeneous applications) [6]. Their topologies can present two types of multiple paths between hosts: uniform and non-uniform ones. ECMP is the standard technique used for splitting traffic across equal-cost (uniform) paths. Nonetheless, it cannot fully utilize the available capacity in these multiple paths [59]. Non-uniform multiple paths, in turn, complicate the problem, as mechanisms must take more factors into account (i.e., path latency and current load). There are some proposals in the literature [46, 53, 57, 58] to address this issue, but they either cannot achieve the desired response times (e.g., Hedera) [86] or are developed for specific architectures (e.g., PSSR for SecondNet). Chiesa et al. [87] have taken an initial approach towards analyzing ECMP and propose algorithms for improving its performance. Nevertheless, further investigation is required for routing traffic across both uniform and non-uniform parallel paths, considering not only tree-based topologies, but also newer proposals such as random graphs [7, 13]. This investigation should lead to novel mechanisms and protocols that better utilize available capacity in DCNs (e.g., eliminating bottlenecks at level-0 links in DCell).

4.8 SUMMARY

In this chapter, we have presented basic foundations of datacenter networks and relevant standards, as well as recent proposals in the literature that address limitations of current mechanisms. We began by studying network topologies in Section 4.2. First, we examined the typical topology utilized in today's datacenters, which consists of a multi-rooted tree with path diversity. This topology is employed by providers to allow rich connectivity with reduced operational costs. One of its drawbacks, however, is the lack of full bisection bandwidth, which is the main motivation for proposing novel topologies. We used a three-class taxonomy to organize the state-of-the-art datacenter topologies: switch-oriented, hybrid switch/server and server-only topologies. The distinct characteristic is the use of switches and/or servers: switches only (Fat-Tree, VL2 and OSA), switches and servers (DCell and BCube) and only servers (CamCube) to perform packet routing and forwarding in the network.

These topologies, however, usually present rigid structures, which hinders incremental network expansion (a desirable property for the ever-growing cloud datacenters). Therefore, we took a look at network expansion strategies (Legup, Rewire and Jellyfish) in Section 4.3. All of these strategies have the goal of improving bisection bandwidth to increase agility (the ability to assign any VM of any application to any server). Furthermore, the design of novel topologies and expansion strategies must consider the nature of traffic in DCNs. In Section 4.4, we summarized recent measurement studies about traffic and discussed some proposals that deal with traffic management on top of a DCN topology.

Then, we discussed routing and addressing in Sections 4.5 and 4.6, respectively. Routing was divided in two categories: layer-3 and layer-2. While layer-3 routing typically employs ECMP and VLB to utilize the high-capacity available in DCNs through the set of multiple paths, layer-2 routing uses the spanning tree protocol. Despite the benefits, these schemes cannot efficiently take advantage of multiple paths. Consequently, we briefly examined proposals that deal with this issue (Hedera, PSSR, SPAIN and Portland). Addressing, in turn, is performed by using LAN technologies, which does not provide robust isolation and flexibility for tenants. Towards solving these issues, we examined the proposal of a new standard (VXLAN) and commercial solutions developed by Amazon (VPC) and Microsoft (Hyper-V). Furthermore, we discussed proposals in the literature that aim at separating name and locator (VL2 and Crossroads) and at allowing full address space virtualization (NetLord).

Finally, we analyzed open research challenges regarding datacenter networks: (i) the need to design more efficient DCNs with heterogeneous sets of devices, while considering optimality; (ii) strategies for incrementally expanding networks with general topologies; (iii) network schemes with strict guarantees and predictability for tenants, while allowing work-conserving sharing to increase utilization; (iv) address flexibility to make the migration of applications to the cloud easier; and (v) mechanisms for load balancing traffic across different multiple parallel paths (using all available capacity).

Having covered the operation and research challenges of intra-datacenter networks, the next three chapters inside the networking and communications part discuss the following subjects: inter-datacenter networks, an important topic related to cloud platforms composed of several datacenters (e.g., Amazon EC2); the emerging paradigm of SDN, its practical implementation (OpenFlow) and how these can be applied to intra- and inter-datacenter networks to provide fine-grained resource management; and mobile cloud computing, which seeks to enhance capabilities of resource-constrained mobile devices using cloud resources.

REFERENCES

1. Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM*, 2011.
2. Alan Shieh, Srikanth Kandula, Albert Greenberg, Changhoon Kim, and Bikas Saha. Sharing the data center network. In *USENIX NSDI*, 2011.

3. Dennis Abts and Bob Felderman. A guided tour of data-center networking. *Communication of the ACM*, 55(6):44–51, 2012.
4. Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM*, 2008.
5. Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM*, 2008.
6. Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: A scalable and flexible data center network. In *ACM SIGCOMM*, 2009.
7. Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. High throughput data center topology design. In *USENIX NSDI*, 2014.
8. Jian Guo, Fangming Liu, Xiaomeng Huang, John C.S. Lui, Mi Hu, Qiao Gao, and Hai Jin. On efficient bandwidth allocation for traffic variability in datacenters. In *IEEE INFOCOM*, 2014a.
9. Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *ACM IMC*, 2010.
10. Nathan Farrington, Erik Rubow, and Amin Vahdat. Data Center Switch Architecture in the Age of Merchant Silicon. In *IEEE HOTI*, 2009.
11. Andrew R. Curtis, Tommy Carpenter, Mustafa Elsheikh, Alejandro Lopez-Ortiz, and S. Keshav. Rewire: An optimization-based framework for unstructured data center network design. In *IEEE INFOCOM*, 2012.
12. Andrew R. Curtis, S. Keshav, and Alejandro Lopez-Ortiz. Legup: Using heterogeneity to reduce the cost of data center network upgrades. In *ACM Co-NEXT*, 2010.
13. Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In *USENIX NSDI*, 2012.
14. Yang Liu and Jogesh Muppala. Fault-tolerance characteristics of data center network topologies using fault regions. In *IEEE/IFIP DSN*, 2013.
15. Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. A cost comparison of datacenter network architectures. In *ACM Co-NEXT*, 2010.
16. Charles Clos. A Study of non-blocking switching networks. *BellSystem Technical Journal*, 32:406–424, 1953.
17. Kai Chen, Ankit Singlay, Atul Singhz, Kishore Ramachandranz, Lei Xuz, Yueping Zhangz, Xitao Wen, and Yan Chen. Osa: An optical switching architecture for data center networks with unprecedented flexibility. In *USENIX NSDI*, 2012.
18. Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. BCube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*, 2009.
19. Hussam Abu-Libdeh, Paolo Costa, Antony Rowstron, Greg O'Shea, and Austin Donnelly. Symbiotic routing in future data centers. In *ACM SIGCOMM*, 2010.
20. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, 2001.
21. Paolo Costa, Thomas Zahn, Ant Rowstron, Greg O'Shea, and Simon Schubert. Why should we integrate services, servers, and networking in a data center? In *ACM WREN*, 2009.
22. Transparent Interconnection of Lots of Links (TRILL): RFCs 5556 and 6325, 2013. Available at: http://tools.ietf.org/rfc/index. Accessed November 20, 2014.

23. Changhoon Kim, Matthew Caesar, and Jennifer Rexford. Floodless in seattle: A scalable ethernet architecture for large enterprises. In *ACM SIGCOMM*, 2008.
24. Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In *USENIX NSDI*, 2010.
25. Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *USENIX NSDI*, 2011.
26. Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
27. Renato Recio. The coming decade of data center networking discontinuities. ICNC, August 2012. keynote speaker.
28. Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. MicroTE: Fine grained traffic engineering for data centers. In *ACM CoNEXT*, 2011.
29. Peter Bodík, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A. Maltz, and Ion Stoica. Surviving failures in bandwidth-constrained datacenters. In *ACM SIGCOMM*, 2012.
30. Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: Measurements & analysis. In *ACM IMC*, 2009.
31. Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *IEEE INFOCOM*, 2010.
32. Chi H. Liu, Andreas Kind, and Tiancheng Liu. Summarizing data center network traffic by partitioned conservative update. *IEEE Communications Letters*, 17(11):2168–2171, 2013.
33. Meng Wang, Xiaoqiao Meng, and Li Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *IEEE INFOCOM*, 2011.
34. Alexandrm-Dorin Giurgiu. Network performance in virtual infrastrucures, February 2010. Available at: http://staff.science.uva.nl/~delaat/sne-2009-2010/p29/presentation.pdf. Accessed November 20, 2014.
35. Dave Mangot. Measuring EC2 system performance, May 2009. Availabel at: http://bit.ly/48Wui. Accessed November 20, 2014.
36. Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: Observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment*, 3(1–2):460–471, 2010.
37. Guohui Wang and T. S. Eugene Ng. The impact of virtualization on network performance of amazon ec2 data center. In *IEEE INFOCOM*, 2010.
38. Haiying Shen and Zhuozhao Li. New bandwidth sharing and pricing policies to achieve A win-win situation for cloud provider and tenants. In *IEEE INFOCOM*. 2014.
39. Eitan Zahavi, Isaac Keslassy, and Avinoam Kolodny. Distributed adaptive routing convergence to non-blocking DCN routing assignments. *IEEE Journal on Selected Areas in Communications*, 32(1):88–101, 2014.
40. Nick G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merive. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM*, 1999.
41. Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C. Snoeren. Cloud control with distributed rate limiting. In *ACM SIGCOMM*, 2007.

42. Joe W. Jiang, Tian Lan, Sangtae Ha, Minghua Chen, and Mung Chiang. Joint VM placement and routing for data center traffic engineering. In *IEEE INFOCOM*, 2012.
43. Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Compuer. Communication Review*, 38:17–29, 2008.
44. Vinh The Lam, Sivasankar Radhakrishnan, Rong Pan, Amin Vahdat, and George Varghese. Netshare and stochastic netshare: Predictable bandwidth allocation for data centers. *ACM SIGCOMM CCR*, 42(3), 2012.
45. Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. FairCloud: Sharing the network in cloud computing. In *ACM SIGCOMM*, 2012.
46. Chuanxiong Guo, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. SecondNet: A data center network virtualization architecture with bandwidth guarantees. In *ACM CoNEXT*, 2010.
47. Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *USENIX WIOV*, 2011.
48. Di Xie, Ning Ding, Y. Charlie Hu, and Ramana Kompella. The only constant is change: Incorporating time-varying network reservations in data centers. In *ACM SIGCOMM*, 2012.
49. Hitesh Ballani, Keon Jang, Thomas Karagiannis, Changhoon Kim, Dinan Gunawardena, and Greg O'Shea. Chatty tenants and the cloud network sharing problem. In *USENIX NSDI*, 2013a.
50. Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Changhoon Kim, and Albert Greenberg. EyeQ: Practical network performance isolation at the edge. In *USENIX NSDI*, 2013.
51. Daniel Stefani Marcon, Rodrigo Ruas Oliveira, Miguel Cardoso Neves, Luciana Salete Buriol, Luciano Paschoal Gaspary, and Marinho Pilla Barcellos. Trust-based Grouping for Cloud Datacenters: Improving security in shared infrastructures. In *IFIP Networking*, 2013.
52. Lucian Popa, Praveen Yalagandula, Sujata Banerjee, Jeffrey C. Mogul, Yoshio Turner, and Jose Renato Santos. ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing. In *ACM SIGCOMM*, 2013.
53. Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *USENIX NSDI*, 2010.
54. C. Hopps. Analysis of an equal-cost multi-path algorithm, 2000. RFC 2992.
55. Albert Greenberg, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Towards a next generation data center architecture: Scalability and commoditization. In *ACM PRESTO*, 2008.
56. Leslie G. Valiant and Gordon J. Brebner. Universal schemes for parallel communication. In *ACM STOC*, 1981.
57. Zhiyang Guo, Jun Duan, and Yuanyuan Yang. On-line multicast scheduling with bounded congestion in Fat-Tree data center networks. *IEEE Journal on Selected Areas in Communications*, 32(1):102–115, 2014b.
58. Wen-Kang Jia. A scalable multicast source routing architecture for data center networks. *IEEE Journal on Selected Areas in Communications*, 32(1):116–123, 2014.

59. Sivasankar Radhakrishnan, Malveeka Tewari, Rishi Kapoor, George Porter, and Amin Vahdat. Dahu: Commodity switches for direct connect data center networks. In *ACM/IEEE ANCS*, 2013.
60. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38:69–74, 2008.
61. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture, 2001. RFC 3031.
62. Dan Li, Mingwer Xu, Ying Liu, Xia Xie, Yong Cui, Jingyi Wang, and Gihai Chen. Reliable multicast in data center networks., 2014. IEEE Transactions Computers, 63: 2011-2024, 2014.
63. 802.1D - MAC Bridges, 2013. Available at: http://www.ieee802.org/1/pages/802.1D.html. Accessed November 20, 2014.
64. Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. PortLand: A scalable fault-tolerant layer 2 data center network fabric. In *ACM SIGCOMM*, 2009.
65. 802.1s - Multiple Spanning Trees, 2013. Available at: http://www.ieee802.org/1/pages/802.1s.html. Accessed November 20, 2014.
66. 802.1ax - Link Aggregation Task Force, 2013. Available at: http://ieee802.org/3/axay/. Accessed November 20, 2014.
67. 802.1Q - Virtual LANs, 2013. Available at: http://www.ieee802.org/1/pages/802.1Q.html. Accessed November 20, 2014.
68. Understanding Multiple Spanning Tree Protocol (802.1s), 2007. Available at: http://www.cisco.com/en/US/tech/tk389/tk621/technologies_white_paper09186a0080094cfc.shtml. Accessed November 20, 2014.
69. Md. Faizal Bari, Raonf Boutaba, Rafael Esteves, Lisandro Z. Granville, Maxim Podlesny, Md. Golam Rabbani, Qi Zhang, and Mohamed F. Zhani. Data center network virtualization: A survey. *IEEE Communications Surveys Tutorials*, 15(2):909–928, 2013.
70. Vijay Mann, Ailkumar Vishnoi, Kalapriya Kannan, and Shivkumar Kalyanaraman. CrossRoads: Seamless VM mobility across data centers through software defined networking. In *IEEE/IFIP NOMS*, 2012.
71. Jayaram Mudigonda, Praveen Yalagandula, Jeff Mogul, Bryan Stiekes, and Yanick Pouffary. NetLord: A scalable multi-tenant network architecture for virtualized datacenters. In *ACM SIGCOMM*, 2011.
72. Brent Stephens, Alan Cox, Wes Felter, Colin Dixon, and John Carter. PAST: Scalable ethernet for data centers. In *ACM CoNEXT*, 2012.
73. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks, 2013. Available at: http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-02. Accessed November 20, 2014.
74. Amazon Virtual Private Cloud, 2013. Available at: http://aws.amazon.com/vpc/. Accessed November 20, 2014.
75. Microsoft Hyper-V Server 2012, 2013a. Available at: http://www.microsoft.com/en-us/server-cloud/hyper-v-server/. Accessed November 20, 2014.
76. Hyper-V Architecture and Feature Overview, 2013b. Available at: http://msdn.microsoft.com/en-us/library/dd722833(v=bts.10).aspx. Accessed November 20, 2014.

77. Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *USENIX OSDI*, 2010.
78. Yan Zhang and Nirwan Ansari. On architecture design, congestion notification, tcp incast and power consumption in data centers. *Communications Surveys Tutorials, IEEE*, 15(1):39–64, 2013.
79. Facebook to Expand Prineville Data Center, 2010. Available at: https://www.facebook.com/notes/prineville-data-center/facebook-to-expand-prineville-data-center/411605058132. Accessed November 20, 2014.
80. Tad Andersen. Facebook's Iowa expansion plan goes before council, 2014. Available at: http://www.kcci.com/news/facebook-just-announced-new-expansion-plan-in-iowa/25694956#!0sfWy. Accessed November 20, 2014.
81. David Cohen. Facebook eyes expansion of oregon data center, 2012. Available at: http://allfacebook.com/prineville-oregon-data-center-expansion_b97206. Accessed November 20, 2014.
82. John Rath. Facebook Considering Asian Expansion With Data Center in Korea, 2013. Available at: http://www.datacenterknowledge.com/archives/2013/12/31/asian-expansion-has-facebook-looking-at-korea/. Accessed November 20, 2014.
83. Ryan Shea, Feng Wang, Haiyang Wang, and Jiangchuan Liu. A deep investigation into network performance in virtual machine based cloud environment. In *IEEE INFOCOM*. 2014.
84. Katrina LaCurts, Shuo Deng, Ameesh Goyal, and Hari Balakrishnan. Choreo: Network-aware task placement for cloud applications. In *ACM IMC*, 2013.
85. Fei Xu, Fangming Liu, Hai Jin, and A.V. Vasilakos. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE*, 102(1):11–31, 2014.
86. Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: Scaling flow management for high-performance networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pp. 254–265, New York, 2011. ACM. Available at: http://doi.acm.org/10.1145/2018436.2018466. Accessed November 20, 2014.
87. Marco Chiesa, Guy Kindler, and Michael Schapira. Traffic engineering with equal-cost-multiPath: an algorithmic perspective. In *IEEE INFOCOM*, 2014.