Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

A Cost-Efficient QoS-Aware Analytical Model of Future Software Content Delivery Networks

Beatriz Otero*[1] | Eva Rodríguez[1] | Otilio Rojas[2,3] | Javier Verdú[1] | Juan José Costa[1] | Manuel Alejandro Pajuelo[1] | Ramón Canal[1]

[1]Computer Architecture Department, Universitat Politècnica de Catalunya, Barcelona, Spain
[2]Barcelona Supercomputing Center, Barcelona, Spain
[3]Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela

Correspondence
*Beatriz Otero, C/Jordi Girona 1-3, Campus Nord, Building C6-204, Universitat Politècnica de Catalunya, 08034, Barcelona, Spain. Email: botero@ac.upc.edu

Present Address
C/Jordi Girona 1-3, Campus Nord, Building C6-204, Universitat Politècnica de Catalunya, 08034, Barcelona, Spain

Summary

Freelance, part-time, work-at-home, and other flexible jobs are changing the concept of workplace, and bringing information and content exchange problems to companies. For instance, the emerging case of companies with mobile workers with different types of corporate devices. Geographically spread corporations may use remote distribution of software and data to attend employees' demands, by exploiting emerging delivery technologies. In this context, cost-efficient software distribution is crucial to allow business evolution and make IT infrastructures more agile. On the other hand, container based virtualization technology is shaping the new trends of software deployment and infrastructure design. We envision current and future enterprise IT management trends evolving towards container based software delivery over Hybrid CDNs. This paper presents a novel cost-efficient Quality-of-Service (QoS) aware analytical model and a Hybrid CDN-P2P architecture for enterprise software distribution. First, we test this model for a wide range of companies ranging from multinationals to SMEs, by using the network simulation tool PeerSim. These experiments emulate content distribution carry out by the real commercial delivery networks KeyCDN, MaxCDN, CDN77 and BunnyCDN, operating under various industrial hypothetical scenarios. We conclude that MaxCDN offers the best cost-QoS trade-off. Later, we include our CDN-P2P architecture into this experimental setting, and use CDN-based results as reference. This study aims at assessing if building a proprietary-owned hybrid CDN-P2P is the most cost-effective solution in a long-term. Simulation results show overall economic savings between 5% and 20%, compared to just hiring resources from a commercial CDN, while guaranteeing satisfactory QoS levels in terms of response times and number of served requests.

KEYWORDS:
cost-efficiency, hybrid architecture, containers, P2P, CDN

1 | INTRODUCTION

Software distribution, especially among employees of the same company, may be open for optimization in terms of cost reduction and making IT infrastructures of emerging SMEs more agile. Similarly, well-consolidated companies with remotely distributed offices, could be potential optimization scenarios of this sort. From the beginning companies pushed for server -based distribution software[1]. As software development evolves new challenges appear, such as mobile workers with different types of corporate devices, or cases where *bring your own device* (BYOD) is allowed, and even also corporate-owned personally enabled (COPE) trends. All of these representing security and management challenges for IT[2].

Cloud computing has proven its effectiveness, not only in reducing the execution time of workloads with high computing and storage requirements, but has also enabling the deployment of a large number of cloud services for enterprises and users in a simple and scalable way. The key to this deployment lies in the increasing use of application containers[3,4]. Basically, an application container (also known as light-weight virtual machine or application virtualizer) is a file that contains an application along with all its dependencies and an execution environment (usually a container loader). The recent advent of container-based virtualization technology is shaping the new trends of software deployment and infrastructure design[5]. In fact, common problems of corporate software, like legacy issues, are currently addressed by the use of container-based technology (e.g. Docker, Avanade, Cisco, Microsoft, Hewlett Packard Enterprise partnership[6]). In no case, an application container emulates the behaviour of a computer's hardware, since it is neither designed to create a standard application execution environment, nor to facilitate the cross-execution of operating systems. An application container only virtualizes the file system in which the application operates along with the name-space of the operating system for that application. In addition, the execution of an application in a container is native, which means that hardware is never virtualized. These container features highly facilitate the deployment of cloud services.

Application containers are displacing the use of more traditional solutions based on virtual machines such as VirtualBox or VMWare. The container's main advantage is the reduction of resources needed for deployment[7]. Specifically, the execution of a containerized application is native and does not incorporate any virtualization layer, and then it reduces processor consumption and the amount of memory required to run this application. In addition, only the application code and data are actually stored in a container, and there is no need to also store the operating system image, as required for virtual machines to run same application. This significantly reduces both the amount of storage and the bandwidth needed to deploy the container. This further makes the application maintenance, especially its updates, less expensive to perform and deploy when using containers.

Aforementioned features make containers ideal for mass deployment of elastic infrastructure services in the cloud, so large providers of cloud infrastructure are developing their own container systems. The popularization of cloud services has led large companies to build infrastructures based on these technologies to provide container deployment services for other companies or individuals. Examples of these infrastructures are Microsoft Azure or Amazon Web Services. The fact of being able to build such container deployments, facilitates the management of the contracted infrastructure, since it avoids all the management of creation, instantiation, deployment and maintenance of virtual machines.

Several solutions and frameworks work on top of proprietary or outsourced servers, although scalable designs include Content Delivery Networks (CDN). CDN emerge as solutions to the current problem of a centralized web, unable to guarantee low response time and minimum loss of information, when moving the content of the information to the end users. Although, some studies have recently reported CDN throughput reduction during peak hours because of high demands[8,9,10], these networks have become the emerging backbone for content distribution[11,12]. To alleviate the trafic pressure from peak demands, CDN architectures have being enhanced by using P2P networks to assist CDN servers, and thereby solve scalability and costs issues of traditional CDN[13,14]. Basically, delays on content distribution to end users can be exclusively reduced by a CDN, at the cost of deploying a large number of network edge cache servers. The deployment cost can be reduced or totally avoided by considering P2P networks. However, these architectures also spend a significant amount of network bandwidth, and the departure of the P2P nodes may lead to a reduction of QoS. A hybrid distribution service that nicely couples CDN and P2P can offer a high content-delivery performance and achieve their complementary advantages, as easy management and high QoS, and also the low running costs and high scalability of P2P networks[15,16,17]. The main strength of Hybrid CDNs is that they offer low priced high-quality services.

Recently, some works[18,19] fully exploit the advantages of CDN-P2P architectures to improve content delivery performance, reduce costs and ease network congestion. However, these works do not include an economic study about the cost of using this technology, to determine in which cases using CDN-P2P infrastructures is more convenient. Although CDN services have a huge worldwide demand because of their usefulness for companies, several important corporates, like Netflix, Comcast, Apple, and Facebook among others, have started to build their own private CDN[20]. This strategic decision involves a huge initial investment, but higher a predicted cost reduction. However, it is indeed challenging to design an optimal and cost-aware proprietary CDN. There are multiple relevant factors that impact the infrastructure design in order to guarantee a required QoS level. There are studies about workload characteristics and performance metrics that allow QoS analyses, but unfortunately, they are mainly focused on common multimedia delivery[21,22,23]. To our knowledge, there is an important lack of studies and analyses focused on the specifications and performance needs of CDNs for enterprise software distribution. In this work, we propose the usage of a CDN-P2P architecture for this purpose, and develop a model, considering relevant parameters such as demand for content, bandwidth and storage capacity, for cost benefit analysis, as in the recent works above mentioned. By means of this proposed model, enterprises will be able to estimate usage or acquisition costs of an infrastructure for software delivery among their employees, or among their clients, that accounts for these crucial parameters.

We envision current and future enterprise IT management trends towards container-based software delivery over CDNs or Hybrid CDNs. However, SMEs and large companies present alternative or different sets of problems and challenges, that makes non-trivial finding the best best ways to build or outsource CDN services. On the one hand, cloud providers offer cost-efficient hosting of resources and delivery services, avoiding the need of owning a infrastructure and its potential investments on acquisition and maintenance, taking advantage of the pay-as-you-go strategy. Cloud-based infrastructures aim at providing efficient content delivery services in terms of QoS, scalability and resource efficiency. Cloud vendors lease resources from one or more cloud providers for a cheaper hosting and deployment of applications, that can scale based on the number of user requests.

There are two alternative considerations that may lead to building a hybrid infrastructure. First, most CDNs have prices suitable for multimedia contents, except for the few of them especially focused on software distribution. Second, workload characteristics and corporate needs may present stringent requirements, like huge delivery of software packages and content protection. This paper presents a novel cost-efficient QoS aware analytical model based on a CDN-P2P architecture for software content delivery, which evaluates different industrial scenarios, considers different CDN software providers, and finally determines if such architecture is cost-effective for software distribution. For a given company, this model estimates installation and maintenance costs taking into account relevant parameters, such as the content size, transmission rate, number of users, etc. It is worthy to noting that our model also determines if building a proprietary-owned CDN is the most cost-effective solution in a long-term. Moreover, our work develops a new delivery framework for enterprise software distribution. To this end, we first present the initial and essential stage of model definition, that requires stating workload characterization and performance metrics for enterprise software delivery. To the best of our knowledge, this is the first throughout analysis of the needs and constraints for software delivery. Moreover, we perform an empirical study of current commercial software delivery CDNs in terms of costs, storage, performance, and reliability. In addition, we define a hybrid architecture that combines CDN and P2P delivery and exhibits three main benefits: (i) high QoS of CDNs, (ii) low priced software delivery and, (iii) high scalability of P2P networks. Finally, we also validate this model against real service and infrastructure providers under different case studies, representative of several SME and big company scenarios. Results confirm the model accuracy and efficiency by taking optimal decisions when designing future enterprise software distribution.

In summary, the key contributions of this paper are:

- Definition of essential features characterizing software distribution, specifically for container-based delivery.
- Review and analysis of existing software delivery CDNs considering costs, storage, performance, reliability and delivery acceleration.
- Definition of a Hybrid CDN-P2P architecture for low priced high QoS (in terms of response time deployment) software applications.
- Definition of a novel cost-efficient QoS aware analytical model that minimizes deployment costs, and satisfies users' QoS in terms of response time and the number of served requests.
- Identification and study of realistic scenarios for model application.
- Model analysis and assessment in all scenarios under consideration.

The rest of this paper is structured as follows. Section 2 discusses the related work. Section 3 presents an analysis of software distribution networks, that first characterizes workload and performance metrics, and then studies current commercial CDNs. Section 4 proposes a Hybrid CDN-P2P architecture for cost-efficient QoS deployment of software applications. Section 5 proposes our cost-efficient QoS-aware model. Section 6 presents tested scenarios and carried out empirical experiments, while Section 7 analyzes all obtained results using the PeerSim simulator. Finally, section 8 presents our conclusions and poses future research directions.

2 | RELATED WORK

This paper proposes a novel cost-analytic model on delivery networks for software distribution. The proposed model will provide companies the best solution in terms of cost and QoS (response time) when externalizing their services and resources. In the literature, we can find different works proposing economic models for minimizing rental costs in Cloud-based CDNs, as well as in Hybrid CDN-P2P networks. Below, we describe relevant works focused on both CDNs and Hybrid CDN-P2P, some of them involving cost modelization, and highlight main differences with respect to the proposed model in this paper.

Large scale and fast dissemination of software updates to millions of Internet users is becoming crucial to offer updated services and applications. In this way, to study the way in which the updates are carried out is really useful to understand software distribution. For Windows update, for instance, a well designed implementation of a fast and effective patch dissemination system has been necessary[24]. In that work, Gkantsidis and co-authors also analyse an alternative patch delivery strategy such as caching and peer-to-peer for software distribution. That paper demonstrates that P2P architectures have a great potential for providing fast and effective patch delivery. However, their results are reported for scenarios where many users download few large files or many small files, but such scenarios do not correspond to the content distribution systems devised in this work. In our case, users could download selected contents whenever they wish, and these contents may be the same each time. File sizes can be variable, and even of large size, accounting for the stringest case of many users downloading a lot of large files. As mentioned before, our model and results include the consideration of P2P architectures for the distribution of contents with business purposes, although not exclusively.

More recently,[18] studies the usage of content distribution techniques and their suitability to network infrastructures with the goal of improving service performance. This is still a new research area, with a lot of challenging issues that need to be addressed. In the literature, we also find works that introduce hybrid solutions blending CDN and Peer-to-Peer (P2P) approaches[25,26,27,18,19]. They combine CDNs and P2P networks taking profit of the advantages of both architectures, that is, the easy management and high QoS of CDNs, and the low running costs and high scalability of P2P networks.

Hu, et al.[28] propose different algorithms to rent cloud resources for building CDNs, by caching resources in a way that costs were minimized, but at the same time, guaranteeing that all requests are served. The first algorithm is the Differential Provisioning and Caching algorithm that minimizes rental cost and optimizes content caching. The second algorithm is the Caching and Request Balancing algorithm to dynamically adjust the placement of replicas in CDNs to maximize the rented resources at runtime. Both algorithms were designed to support dynamic demand patterns. Main differences with respect to our algorithm are the nature of the content, user types, and demand patterns. While, Hu and co-authors consider multimedia contents, media consumption, and high varying patterns, we consider containerized software applications, employees of a corporation, and well known (regarding time) demand patterns. It is worth noting that all these factors have a direct impact on model variables.

Other works in the literature present different solutions for replica placement. Sahoo and Glitho propose in[29] an efficient heuristic for the NP-hard replica server placement problem. The heuristic first places replicas on resources, mainly servers with low operational cost, on the Cloud, and then it reduces redundant cloud sites. This heuristic improves existing ones in terms of computation time and operational cost. Alternative replica placement algorithms have been developed for cost minimization[30,31]. These works provide the best solutions for replica placement, an NP-complete problem, based on tree topologies. The later also considers reading, writing, and storage costs. These works are similar to ours for replica placement, but we here also account for economic costs.

Garmehi, et al.[15] propose an economic and efficient replica placement algorithm for content streaming. They use the Economic Mechanism Design Theory to design a recursive hierarchical replica placement mechanism. Advantages of this model become clear when the system load increases significantly, since it is when edge servers replicate content strategically. This model reduces costs to content providers, spending less for delivered traffic, and at the same time improves QoS to end-users. The main difference between[15] and our model is that we minimize rental costs, while serving all requests and satisfying QoS in terms of response time (minimum content delivery time).

Alternative published contributions address auto elasticity algorithms in the Cloud[32,33], both reactive and predictive, where resources are optimized according to user demands. Cerqueira and Solis[34] present an architecture based on containers to promote auto elasticity on a Cloud computing environment. They propose a Proportional-Integral-Derivative (PID) based auto scaler algorithm to optimize resource allocations complying with response time requirements. The PID controller reacts to variations of the system's response time, by changing the number of containers in the load balancing cluster that process web requests. That work is related to the one presented in this paper, since our model also guarantees users' QoS in terms of response time.

But, the main difference is that response time in our model has an impact on the rented resources; while in Cerqueira-Solis' algorithm response time impacts the number of containers deployed.

A recent work[35] presents an economic study of a request-routing and resource-allocation mechanism in CDN-P2P networks. However, this mechanism is used to replicate the content in client-server or P2P modes, and determine the optimum level of end-user's contribution, but not define a cost-efficient QoS aware analytical model to minimizes deployment costs.

3 | ANALYSIS OF SOFTWARE DISTRIBUTION NETWORKS

The state of the art on CDN evaluation metrics reduces to five key metrics[36], focused on common content distribution, such as Web objects, static data, or multimedia content. These metrics are cache hit ratio, saved bandwidth, latency, surrogate server utilization, and reliability. All of them aim at reducing the waiting time of end-users to get access to the requested content. Even though, software distribution presents different requirements to guarantee QoS.

This section reviews the first characterization found in the literature of appropriate features for software distribution, specifically for container-based software. First, we introduce a brief description of workload characteristics. Then, we present the key performance metrics impacting the required QoS. Finally, we study and compare nowadays commercial infrastructures for content delivery.

3.1 | Workload Characteristics and Performance Metrics

This paper envisions future software distribution based on containers. In the beginning, container-based distribution was mainly used for micro service deployments. Recently, they have been also used for delivering applications, as in the case of enterprise software[6]. Container-based software delivery presents particular characteristics, namely:

- Large content sizes. Containerized software applications are much bigger than typical multimedia contents (audio, video, etc.). The size of a containerized application ranges from 5 GB to 40 GB[37], depending on the application.
- Partial downloads. A specific feature of software containers, unlike software images or video games, is that the containerized application can be executed only with about 20% of the total data stored in the container[38]. This significantly reduces the minimum required data to be delivered, in order to guarantee a QoS level, although it remains much higher than the essential for multimedia content delivery, in typical CDNs. For example, Netflix requires from 0.3 GB/hour for a movie in low resolution up to 7 GB/hour for a movie in ultra HD. While, for software containers, the distribution network has to deliver between 1 GB and 8 GB in a few minutes to execute the application. Thus, the partial download characteristic allow the hybrid network only transmitting this 20% through the CDN to provide an acceptable QoS to end users.
- With/without data. Depending on user needs, the container may only comprise the software application, or it may also contain data to be processed by the software. In the latter case, the container contents need to be updated to include the latest files and required data. This feature has a direct impact on the network traffic (just downloading versus bidirectional trafic) and consequent costs.

An alternative set of features relevant for network architecture are user related. In software delivery, users usually are part of the same corporation, unlike users of multimedia content distribution. Thus, it is also necessary have into account the following system parameters:

- Well identified peaks. User requests will be more frequent at the start of a working day. Thus, peaks will be more predictable over time.
- The number of delivery requests will not fluctuate. For companies under normal operation, the number of employees will not significantly vary over time, and therefore, the number of requests for a given application will follow a similar temporal pattern. Unlike the case of multimedia content distribution, where the number of users can vary a lot.
- Requests will be geographically concentrated. Most of requests will take place on those regions where the company has its headquarters. This will limit the location of the points of presence within the CDN.

TABLE 1 Comparison of commercial CDNs for software delivery

		Cost			Delivery
CDN provider	**Network**	**Model**	**Transfer**	**Storage**	**acceleration**
keyCDN	25 data centers	pay-as-you-go (minimum 49 $/year)	\$0.04 GB/month (first 10 TB)... \$0.02 GB/month (above 500 TB)	\$0.047 GB/month (first 250 GB) to \$0.27 GB/month (above 500 GB)	SSD servers Optimized TCP stack
CDN77	32 data centers	pay-as-you-go	\$0.049 GB/month (first 5 TB)... \$0.029 GB (above 440 TB)	First 50 GB for free Next 150 GB costs \$20/montly,..., up to \$295/monthly for 5 TB	Leading-edge latency-based routing
MaxCDN	19 edge locations	fix (monthly or annual packages)	\$9/month (100 GB) up to \$1199/month (25 TB)	-	Anycast DNS traffic routing algorithm and SSD servers
		Custom per-gigabyte princing	\$0.095 (25 TB or under) ... 0.05 (3 PB - 5 PB)		
BunnyCDN	23 data centers	pay-as-you-go (minimun 10\$/year)	\$0.01 GB/month	\$0.01 GB/month	SSD servers Anycast DNS network

Finally, regarding performance metrics, QoS will be measured in terms of response time. In the case of software container deployment, the users' satisfaction will mainly depend on the time needed for the application execution. This time will depend on the delivery time of the complete container, or at least of the 20% of it.

3.2 | Infrastructures for Content Delivery

This section provides an analysis of current commercial infrastructures for software delivery in terms of costs, storage, performance, and delivery acceleration. A comparative summary is provided in Table 1. CDNs are distributed networks built of strategically positioned servers, file storage devices or data centers around the world, which allow fast content delivery since end-users are served with the replica of their closest edge, or Point of Presence (PoP). CDNs improve user experience by reducing the response time, and at the same time, alleviating network traffic. Nowadays, few of them are specialized in software distribution, as KeyCDN[39], MaxCDN[40], CDN77[41], or BunnyCDN[42]. These providers can better optimize the content pricing and caching scheme, when compared to the traditional providers for delivery multimedia content, where a subscription service is hired, instead of on-demand service. As a complementary comparative study, we refer the reader to the works[35,43,35], that evaluate different CDN providers considering a variety of features.

The commercial networks in Table 1 are later used to test our cost-effective model proposed in section 5, which aims at providing companies an overall efficient delivery of containerized software applications with QoS satisfaction, while minimizing costs; that is, a trade-off between costs and QoS in terms of response time.

KeyCDN has been designed to be scalable, in order to provide high availability of contents and to speed up their delivery. This global CDN has been designed to provide high performance, high throughput, and low latency. KeyCDN offers solutions for software, game and application delivery; and for improving website performance. KeyCDN uses the pay-as-you-go model, but clients have to spend at least \$49 per year, otherwise, their credits expire. KeyCDN has 25 data centers around the world. Clients have for free five zones, but if they want to use PoP in other zones they have to pay \$1 extra per zone/month. KeyCDN enables data transfer rates of 40 GBps. Content can be uploaded and stored directly on KeyCDN's storage cluster creating a push zone. The maximum file size is 5GB. The software delivery platform makes faster downloads because servers benefit from an optimized TPC stack and 100% SSD coverage.

CDN77 provides solutions for web site acceleration, video on demand, software distribution, and gaming. It has 32 data centers around the world, in 27 different countries, and uses the pay-as-you-go model based on traffic. Content also can be uploaded and stored, being the first 50 GB for free, with a maximum number of 500.000 files. CDN77 minimizes delivery time exploiting their leading-edge latency-based routing.

MaxCDN also provides solutions for gaming and software distribution. It is specialized in delivering software updates to millions of devices. It offers customized pricing, based on the pay-as-you-model, ranging from \$0.095 (25TB or under) to \$0.05 (3PB-5PB). However, unlike KeyCDN and CDN77, it also offers monthly or annual packages for entrepreneur or professional users, depending on their needs. Its network is made up of 19 edge locations, strategically placed, and accelerated by the Anycast DNS traffic routing algorithm and SSD servers.

BunnyCDN is a low-cost fast CDN with 23 data centers around the world. It offers solutions for software and content delivery, web site acceleration, and CDN cloud storage. It guarantees latencies less than 30ms. All its servers also are powered by SSD technology, and they have designed an anycast DNS network to provide users the best possible network route. It offers two different solutions: the premium tier for high-performance solutions and the volume tier. The latter uses a smaller set of high-performance PoPs.

From above analysis, we can conclude that CDN77 is the provider with more data centers, which has a direct impact on the network latency. All of them, except MaxCDN, only offer the pay-as-you-go model. Finally, all these providers supply solutions for delivery acceleration based on SSD servers and traffic routing algorithms.

4 | SOFTWARE DELIVERY ARCHITECTURE: HYBRID CDN-P2P ARCHITECTURE

In the last years, Hybrid CDN-P2P content distribution systems[15] have received increasing attention in the research community, as well as in the industry, especially for video streaming. The hybrid CDN-P2P approach is promising to achieve a trade-off between QoS (in terms of deployment time) and distribution costs. This paper proposes a Hybrid CDN-P2P distribution system for software deployment, in particular for containerized software applications. The proposed architecture will profit off the advantages of CDN servers and P2P networks, as in multimedia scenarios. More specifically, it will benefit of the performance in terms of delivery time of CDN servers and of the low distribution costs of P2P networks.

Hybrid systems can be classified into two categories[44], depending on their mix strategy: Peer-assisted CDN and CDN assisted P2P. Most of the current commercial approaches fall into the first category, where the P2P network acts as a complement for the CDN. For example, in video streaming systems, user requests are served by the CDN, and the P2P network mainly improves user experience and alleviates network load stress. The hybrid system that we propose for software delivery, see Figure 1, also fall into this first category, Peer-assisted CDN. User requests also will be served by the CDN, and the critical data will be sent through the CDN architecture, while P2P will alleviate network load stress and reduce costs. In our architecture peer selection also benefits from CDN redirection scheme.

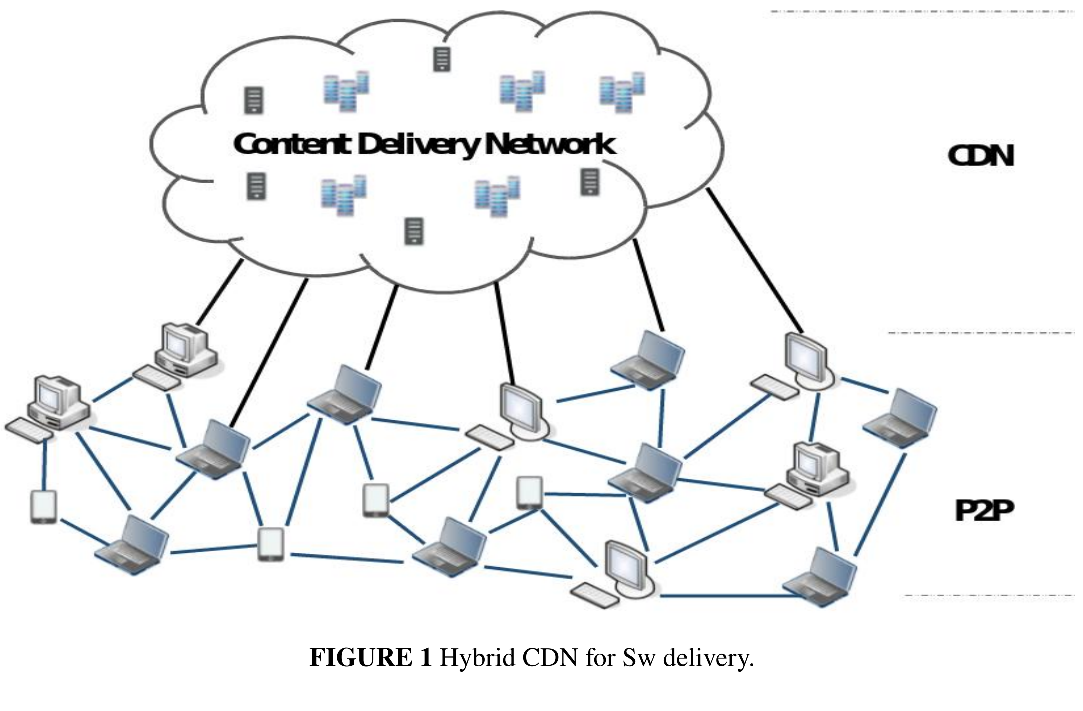

FIGURE 1 Hybrid CDN for Sw delivery.

A key feature of software containers is that containerized applications only need 20% of the total data in containers to be executed[38]. This feature has a direct impact on the design of the hybrid CDN-P2P architecture proposed. This part of the container will be delivered by the CDN servers to guarantee users' QoS since the response time is critical for the user to start to execute the requested application. On the other hand, the remaining 80% of the container will be delivered by both CDN and

P2P networks. Therefore, the CDN approach is advantageous to support the delivery of the critical part of the container, and a P2P approach will provide cost-effective distribution for the remaining modules of the containerized applications.

For the proposed hybrid architecture, the containers' delivery time falls into one of the three stages sketched in Figure 2.

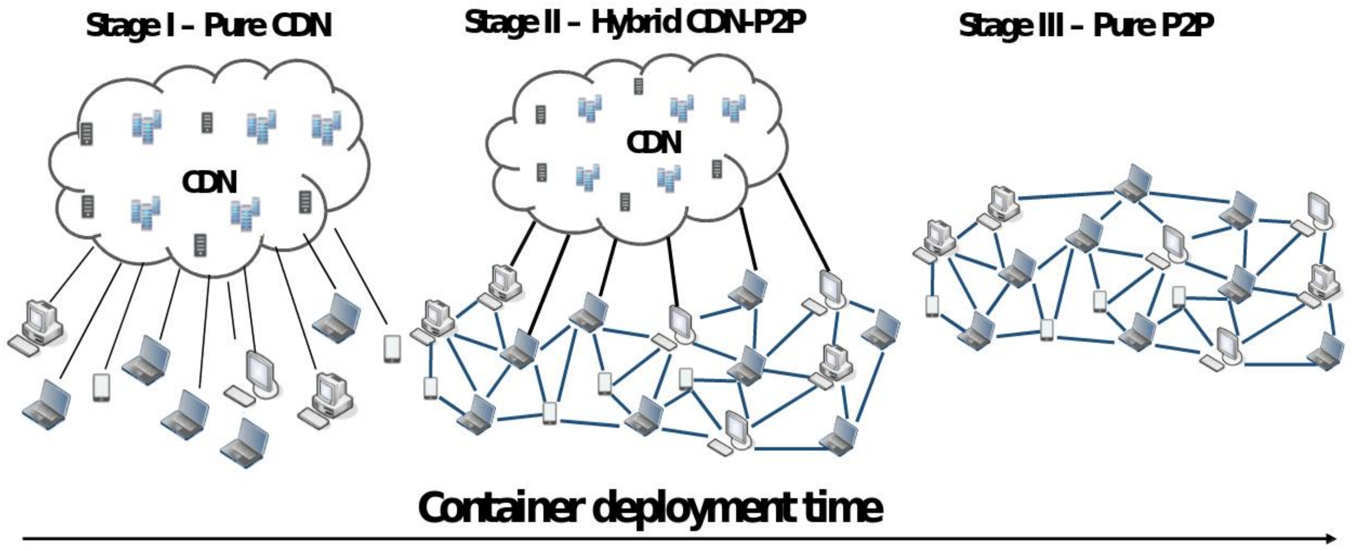

FIGURE 2 Different stages for software containers' deployment.

5 | COST-EFFICIENT QOS-AWARE MODEL

This section proposes an optimization model to be used by a company that decides to outsource its services and resources by renting an infrastructure to reduce operation and capital expenses. This infrastructure should blend Cloud and P2Ps services, taking profit of the delivery costs of P2P and Cloud response time. All model solutions provide elastic resource provisioning to users by means of an automatic resource allocation and load balancing techniques. The model minimizes resource rental costs, satisfying all considered standard demands, with a minimum delivery time. The model formulation uses integer programming with an objective function that minimizes the infrastructure renting costs, and considers certain resource restrictions involving QoS, bandwidth, and storage; all of which must be taken into account when using both infrastructures. The model assumptions are the following:

- Bandwidth costs are lower in P2P than in Cloud, but deployment time is higher (Trade-off between deployment time and cost).
- P2P bandwidth is greater than the chunk size.
- All P2P users are candidates to participate in any request of software distribution.

All contents in the system are divided into multiple equal-sized chunks and delivered. In this work, all contents are of the same size for each request. The system attends all requests and it is assumed a well-known input demand uniformly distributed. The numbers of requests are predictable, then the input demand is not a variable function, and it is a parameter instead.

Each request, issued by a user, attempts to access certain content (a containerized application, the user wants to use). We suppose that user application related data is detached from the container, and stored in a different resource. Then, all user requests of a concrete software application, are served with the same containerized application or corresponding replica. The content, for instance, could be file blocks in file downloading systems.

TABLE 2 Summary of notations and terminology - CDN network.

Parameter	**Definition**
$B_{CS,i}$	Bandwidth capacity of site i
$S_{CS,i}$	Storage capacity of site i
$p^{b}_{CS,i}$	Unit bandwidth rental price of site i
$p^{s}_{CS,i}$	Unit storage rental price of site i
$Q_{CS,i,j}$	Binary parameter indicating whether a request can be routed from location i to site j
$d_{CS,i}$	Input demands for content on location i
Variable	**Definition**
$b_{CS,i}$	Amount of bandwidth rented on location i
$s_{CS,i}$	Amount of storage rented on location i
$r_{CS,i,0}$	Ratio of demands for content routed from location i to the root
$r_{CS,i,j}$	Ratio of demands for content routed from location i to the site j

In order to satisfy the quality of service requirements of end-user requests, we consider a binary parameter Q_{ij} to denote whether requests can be routed from location i to site j. We will assume that serving each request consumes the same amount of resources (bandwidth and storage).

Hereinafter, we will separately present the terms associated with the model objective function, as well as the restrictions for each infrastructure, being the Cloud infrastructure the first presented. Finally, we will propose a model that combines both infrastructures.

5.1 | Cloud Infrastructure

The delivery network is built on resources leased from N cloud sites located across the Internet, and the users reside on M locations. We suppose that $N \ll M$.

We define the number of input demands created at location i as $d_{CS,i}$. Let $b_{CS,i}$ and $s_{CS,i}$ be the amounts of bandwidth capacity and storage capacity that have been rented at localization i. Similarly, let $P^{b}_{CS,i}$ be the unit price of site i to rent bandwidth, and let $P^{s}_{CS,i}$ be the unit storage price to the same site. We denote and as the bandwidth and storage capacities of site i. The notation $r_{CS,i,j}$ is used to denote the probability of requests from location i being directed to site j. We denote the root as site 0 and we use $r_{CS,i,0}$ to denote the fraction demand sent to the root. Demands routed to the root should suffer a high cost $p_{CS,0,b}$ as punishment. Finally, let $Q_{CS,i,j}$ be the parameter that indicates whether a request can be routed from location i to site j. Table 2 summarizes the notations and terminology used in this paper to model the providers' cost in the CDN infrastructure.

As stated above, the main objective is to minimize the total cost of hired resources (bandwidth and storage) to attend all requests, while satisfying the QoS by means of acceptable deployment times. This problem can be formulated as an integer programming problem as follows:

$$\min C_{type} = \min \left(\sum_{i=0}^{N} \left(p^{b_{type}}_{CS,i} * b^{type}_{CS,i} + p^{s_{type}}_{CS,i} * s^{type}_{CS,i} \right) + \sum_{i=0}^{N} \left(p^{b_{type}}_{CS,0} * d_{CS,i} * r^{type}_{CS,i,0} \right) + c^{type}_{f} \right) \tag{1}$$

with c^{type}_{f} the fixed cost of one CDN's provider to $type$ =KeyCDN, CDN77,$\cdots$, BunnyCDN.

s.t

$$b^{type}_{CS,i} \leq B^{type}_{CS,i} \quad i = 0, \cdots, N \tag{2}$$

$$s^{type}_{CS,i} \leq S^{type}_{CS,i} \quad i = 0, \cdots, N \tag{3}$$

$$\sum_{i=0}^{M} \left(r^{type}_{CS,i,j} * d^{type}_{CS,i} \right) \leq b^{type}_{CS,i} \quad j = 0, \cdots, N \tag{4}$$

TABLE 3 Summary of notations and terminology - P2P network.

Parameter	Definition
C_i^t	Total number of chunks that Peer i intends to download from its neighbors at time slot t
T	Chunk size in bytes
$B_{P2P,i}$	Upload bandwidth capacity of Peer i
$S_{P2P,i}$	Storage capacity of Peer i
$p^b_{P2P,i}$	Unit bandwidth rental price for Peer to receive a chunk from one neighbor peer
$p^s_{P2P,i}$	Unit storage rental price for Peer to receive a chunk from one neighbor peer
$SR^C_{i,u\to d}$	Server request indicator. Denotes if Peer i can download chunk c (i. e. its value is 1 if a request is served by the corresponding upstream peer u, and 0 otherwise)
Variable	**Definition**
$b_{P2P,i}$	Amount of bandwidth used on the Peer i for uploading a chunk
$s_{P2P,i}$	Amount of storage used on the Peer i for storing a chunk

$$r_{i,j}^{type}\left(1 - Q_{CS,i,j}^{type}\right) = 0 \quad i = 0, \cdots, M; \quad j = 0, \cdots, N \tag{5}$$

$$r_{CS,i,0}^{type} + \sum_{j=0}^{N} r_{CS,i,j}^{type} = 1 \quad i = 0, \cdots, M \tag{6}$$

where constraints (2), (3) and (4) include the bandwidth limit and disk limit at each site, the constraint (5) guarantees that the QoS requirement can be satisfied, and the constraint (6) guarantees that the total fraction of served requests is 1.

5.2 | P2P Infrastructure

This section presents the cost-efficient model for the P2P infrastructure. We define C_i^t as the total number of chunks that Peer i intends to download from its neighbours at time slot t. We denote T as the chunk size in bytes. Let $B_{P2P,i}$ and $S_{P2P,i}$ be the amounts of bandwidth capacity and storage capacity that Peer i has rented. Similarly, let $p^b_{P2P,i}$ be the unit bandwidth rental price for Peer i for receiving a chunk from one neighbour peer and let $p^s_{P2P,i}$ be the chunk storage price on Peer i. We denote $B_{P2P,i}$ and $S_{P2P,i}$ as the bandwidth and storage capacities of Peer i. The notation $SR^C_{i,u\to d}$ is used to denote if Peer i can download the chunk c (i. e. its value is 1 if a request is served by the corresponding upstream peer u, and 0 otherwise). Table 3 summarizes the notation for parameters and variables of the P2P infrastructure cost-efficient model.

For the definition of the model we will assume the following:

- For all peers, the upload and download bandwidths are the same.
- Once the distribution of a chunk starts, from a neighbour, chunk downloading is completed. The model does not support chunks partially downloaded in a given peer i.
- The first neighbour discovered from peer i is the one chosen to receive the chunk.
- Discovered neighbours of peer i will have available the requested chunks.
- The cost of a delivery chunk will be the same for all neighbours. The model will only consider close neighbours.

Same as for the previous case the cost-effective model the P2P infrastructure is an integer programming model, whose formulation is as follows:

$$\min \sum_{c\in C_i^t} \sum_{u\in N_{P2P,i}} SR^C_{i,u\to d}\left(p^b_{P2P,i} * b_{P2P,i} + p^s_{P2P,i} * s_{P2P,i}\right) \tag{7}$$

with $N_{P2P,i}$ the set of all neighbors of peer i
s.t

$$T * C_i^t \leq b_{P2P,i} \leq B_{P2P,i} \quad i = 0, \cdots, N \tag{8}$$

$$T * C_i^t \leq s_{P2P,i} \leq S_{P2P,i} \quad i = 0, \cdots, N \tag{9}$$

where restriction (8) guarantees that a peer cannot download more content than the maximum allowed by the bandwidth hired. Moreover, it allows peers to download several different chunks at the same time. Finally, restriction (9) guarantees that the storage used by peer i is lower than its whole capacity.

In this way, the model proposed for the hybrid architecture requires optimizing the following objective function:

$$\min\left(\sum_{i=0}^{N}\left(p_{CS,i}^{b_{type}} * b_{CS,i}^{type} + p_{CS,i}^{s_{type}} * s_{CS,i}^{type}\right) + \sum_{i=0}^{N}\left(p_{CS,0}^{b_{type}} * d_{CS,i} * r_{CS,i,0}^{type}\right) + c_f^{type} + \sum_{c\in C_i^t}\sum_{u\in N_{P2P,i}} SR_{i,u\to d}^{C}\left(p_{P2P,i}^{b} * b_{P2P,i} + p_{P2P,i}^{s} * s_{P2P,i}\right)\right) \tag{10}$$

subject to restrictions (2)-(6) and (8)-(9) including the follow constraints:

$$p_{CS,i}^{b_{type}} * b_{CS,i}^{type} > p_{P2P,i}^{b} * b_{P2P,i} \quad i = 0, \cdots, N \tag{11}$$

$$p_{CS,i}^{s_{type}} * s_{CS,i}^{type} \leq p_{P2P,i}^{s} * s_{P2P,i} \quad i = 0, \cdots, N \tag{12}$$

where constraints (11) and (12) assure that P2P unit bandwidth price is cheaper than CS, but the storage unit price is more expensive than CS.

In order to minimize the objective function above, it is necessary, whenever it is feasible (that is, when response times will be acceptable), to deliver contents (software containers) through the P2P architecture.

6 | SCENARIOS AND EMPIRICAL EXPERIMENTS

This section presents the scenarios and experiments carried out to test the effectiveness of our empirical cost-efficient model for commercial CDN providers. This conducted empirical study considers the four CDN providers, specialized in software distribution and reviewed in section 3.2 (see Table 1, for a summary). This study is later used as reference, for simulation results when modeling a hybrid CDN-P2P architecture, and give us an idea of the expected limits or ranges of the container delivery time. Such analyses help us to determine the best option to minimize costs in a concrete real scenario, while guaranteeing the users' QoS.

6.1 | Scenarios

Our scenarios have been designed aiming at encompassing the most representative number of existing corporations, ranging from large and medium-sized multinationals to SMEs. For each of these company types, we have define the number of employees, the employee profiles, and an average number of applications used by each type of employees (see Table 4).

For all our scenarios, three different specific cases can be considered:

1. Users (employees) who download containers with the desired application only once for updating purposes. Each user has his/her own terminal (PC or laptop).
2. Users who download containers every day, because of they work with shared resources, as for instance, technical employees in a lab.
3. Users who download containers, with the required application and personal documents every day, and upload them back to the server, at the end of their working day.

TABLE 4 Corporations' key features.

Scenario	Corporation type	Employee profile	Containerized applications
#1	Multinational (2500-4000 employees)	Administrative (60%)	4 containers/day (average per request) Container size: 5-10 GB
#2		Technical (40%)	5 containers/day (average per request) Container size: 10-20 GB
#3	SME (20-100 employees)	Administrative (60%)	4 containers/day (average per request) Container size: 5-10 GB
#4		Technical (40%)	5 containers/day (average per request) Container size: 10-20 GB

For these three cases, we consider the worst scenario, in terms of the number of requests, given when all employees attempt to download containers at the same time of the day, probably because they start working at the same hour. The conducted study defines different scenarios for each company, depending on the profile of the employees. Those profiles are well differentiated in two different types, either administrative or technical. This is a common situation when companies, especially multinationals, want to solve a concrete problem in one of their departments. Therefore, our model can be applied separately using both profiles, when determining the best choice for a content delivery provider.

6.2 | Empirical Experiments for CDN Providers

These empirical experiments determine the best CDN provider, i.e. the one that minimizes costs and guarantees QoS in terms of response time. Our underlying model is a trade-off between QoS and economic costs, where response time is closely tied to the bandwidth contracted by companies. Based on a recent study of the European Commission about connectivity in the UE[45], and the positives trends on broadband adoption[46], we suppose that SMEs have a connectivity of at least 400 Mbps, and multinationals of at least 1 Gbps. Moreover, the European Commission is providing funding to guarantee to SMEs, as well as to providers of public services and schools, connections of 1 Gbps by 2025. Taking into account these European Commission considerations, all CDN providers can currently guarantee acceptable response times for users, and user satisfaction will remain in the future. Figure 3 sketches delivery times of software containers in the four scenarios considered. For all scenarios, we consider the worst case of simultaneous attempts of downloading a concrete containerized application by all system users. Administrative employees of a multinational will wait between 40 seconds and 3 minutes for a container delivery, depending on the size of desired containerized application. Taking into account that the execution of a containerized application only requires 20% of its whole size[38], these applications only need 35 seconds to start running. Alternatively, response times for the technical staff of this multinational will vary between 35 seconds and 1 minute. In the case of a SME, container deliveries for its administrative staff will spend between 30 seconds and 2 minutes, and between 1.5 and 3 minutes for the case of its technical staff.

The rest of this section focuses on minimizing corporations' rental costs. Figures 4 to 7 sketch the prediction models obtained for each scenario and for each CDN provider under consideration. In all these use cases, we define two models that delimit, for each provider and scenario, the spectrum of lowest and highest costs. The band of lowest cost is drawn with a dashed line, while the one of highest cost is represented with a continuous line. A linear prediction model fitting the empirical data of delivery costs charged by commercial CDN providers (according to the public information on CDN's websites). Note that resulting linear models are highly accurate, since they close follow our empirical data, with R-squared value between 0.6 and 1.0. Prediction models are overlapped with empirical trends for all considered scenarios, as shown in Figures 4 to 7. Labels on these figures, give the analytical expressions of data regression models.

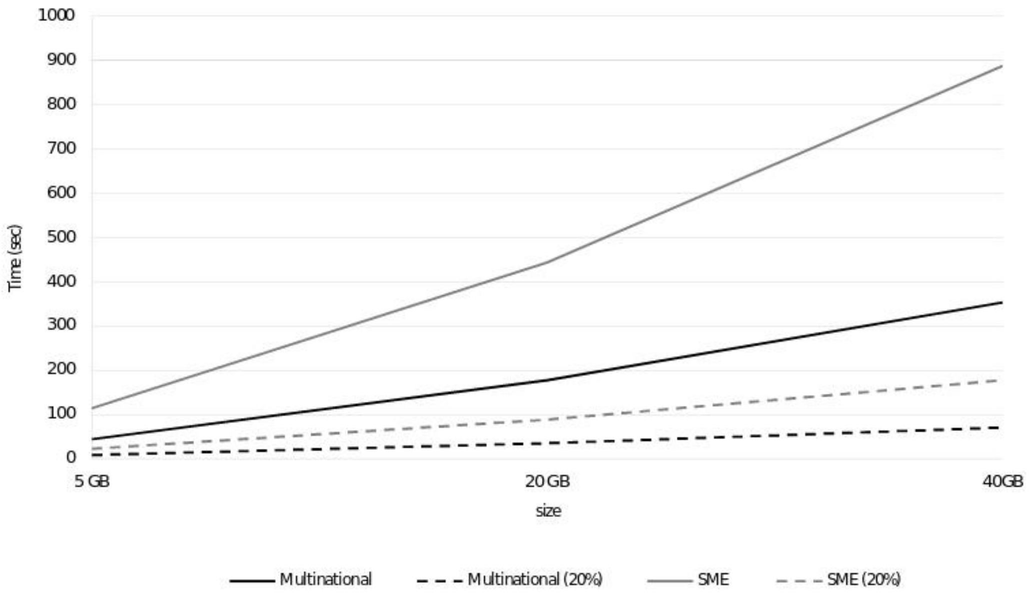

FIGURE 3 Software container delivery times in the four experimental scenarios of section.

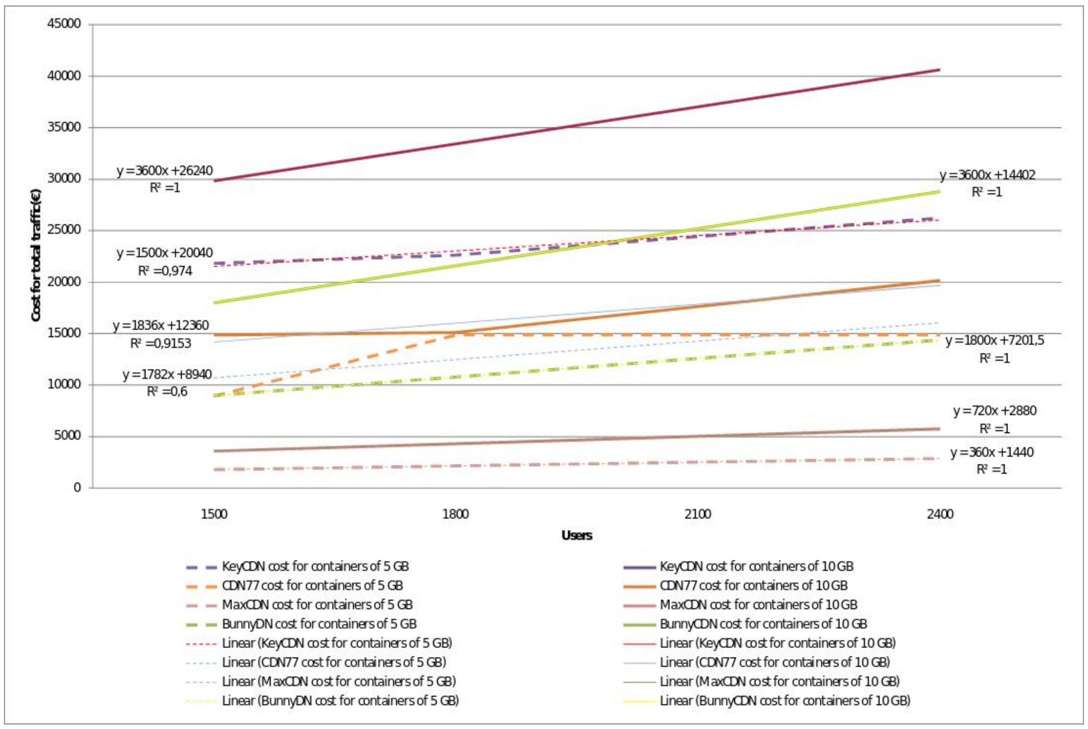

FIGURE 4 Empirical results and prediction models for Scenario # 1. Two containers sizes are considered 5 GB and 10 GB.

Finally, it is important to remark that for all the scenarios under consideration, results in figures 4 to 7 establish that MaxCDN is cheaper than alternative assessed providers, and guarantees QoS levels in terms of response time.

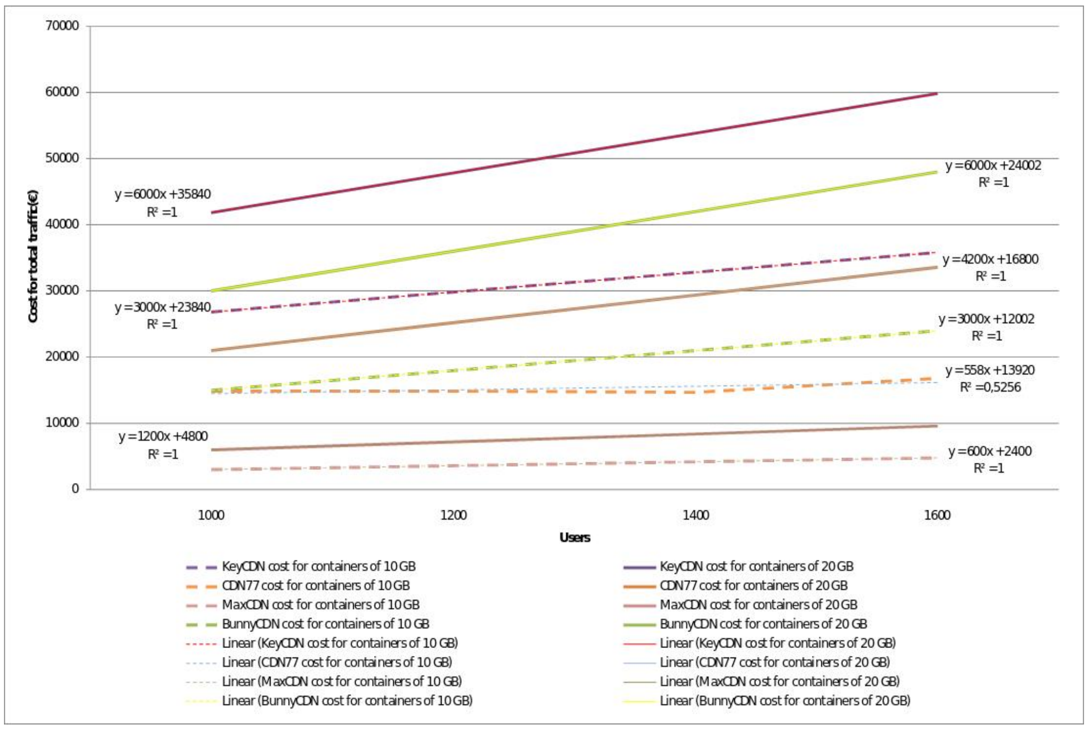

FIGURE 5 Empirical results and prediction models for Scenario # 2. Two containers sizes are considered 10 GB and 20 GB.

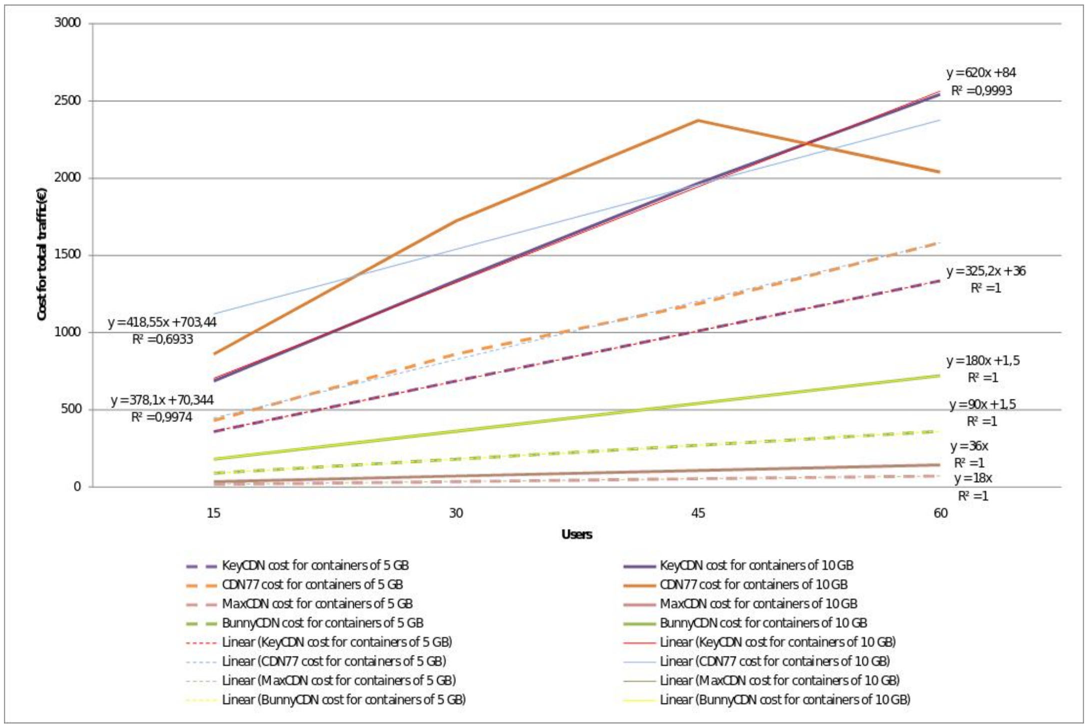

FIGURE 6 Empirical results and prediction models for Scenario # 3 considering all providers and container sizes of 5 GB and 10 GB.

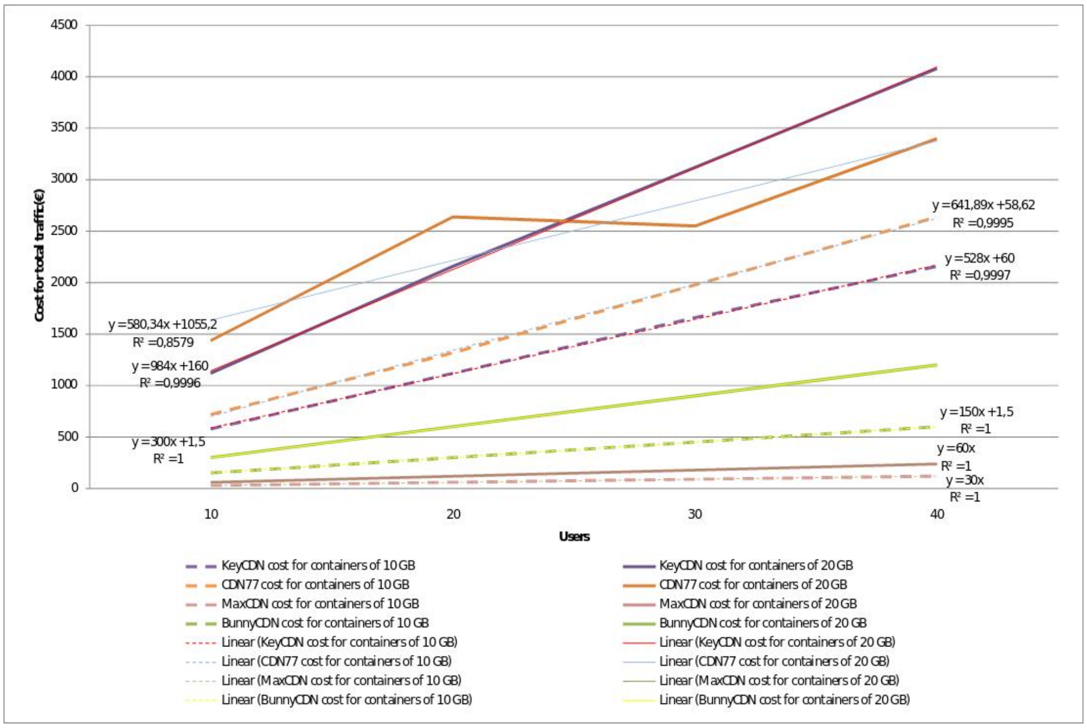

FIGURE 7 Empirical results and prediction models for Scenario # 4 considering all providers and container sizes of 10 GB and 20 GB.

7 | SIMULATION OF A HYBRID CDN-P2P NETWORK

This section describes the simulation software used to model the Hybrid CDN-P2P network proposed in this paper for software distribution, along with all simulation parameters that allow evaluating the effectiveness of our new QoS aware solution. In addition, we also present scenarios and experiments designed to assess the effectiveness of our network and cost-efficient model. Finally, we discuss simulation results and establish a comparison with empirical experiments detailed in section 6.2.

7.1 | Simulation Software and Parameters

In this work, we use PeerSim[44,47] for our CDN-P2P modeling given its reliability and amenability. PeerSim is a scalable simulation environment, which supports dynamic scenarios. It is an open software simulator written in Java programming language and designed to be modular and easy to configure. It enables network modeling, as a list of nodes, each one having a list of protocols. It also gives access to simulation parameters for network configuration and control. Network configuration includes information about network size, node information, control objects, communication protocols, and some additional information for specific requirements, particular to each simulation. PeerSim supports two different simulation models: cycle-based model and event-based model. The former is based on a very simple scheduling algorithm and its main limitation is that users cannot intervene during the simulation process. The later overcomes this limitation enabling the simulation of complex operational networks.

The event-based model is the most suitable to simulate and validate our architecture. However, it requires some source code modifications to adapt network nodes to new functionalities. Specifically, there are two main modifications considered in our implementation are next detailed. First, we categorize nodes into two different types: (i) nodes representing CDN servers and, (ii) peer nodes. CDN nodes have a larger bandwidth than peer nodes, and these CDN bandwidth values have been estimated according to those studied in section 6.2. More specifically, the chosen bandwidth and storage values correspond to those provided by MaxCDN, since this provider offered the minimum costs, and at the same time, guarantees minimum QoS in terms of response times and the number of served requests. On the other hand, Peer bandwidths are, on average, 400 Mbps. This is a typical average value that small and big companies rent from commercial telephone providers. Furthermore, we consider as

TABLE 5 Parameter values used in our simulations.

Parameter	Assumed values
Network size	Employees (scenario depending)
Container size	5-10 GB, 10-20 GB (profile depending)
Chunk size	500 Kb
CDN bandwidth	1 Gbps
Peer bandwidth	400 Mbps
P2P percentage in a buffer	50
Delay between CDN.P2P	20 ms
Delay between P2P-P2P	20-100 ms

the same transfer delay, those requests sent or received between a CDN server and a peer couple, than in case of peer-to-peer transfers. Therefore, we assume the same delay for either uploading or downloading traffic.

The second code modification we made, is defining the percentage of P2P nodes in the PeerSim network configuration file. We add a common buffer to be used by all network nodes, whose size corresponds to the number of slots required to store a chunk during a container deployment. However, there is an additional requirement for this buffer in both cases of a CDN or a peer node. The buffer for CDN nodes has an unlimited capacity, while the buffer used by peer nodes has a finite capacity. This feature is based on our model assumption that all CDN servers are always available to provide all requested chucks to peer nodes, since the provider has enough bandwidth to guarantee it. As we have mentioned in section 4, the first 20% of the container is delivered via CDN, because transmission time is critical to the execution of the containerized software application. Then, the remaining container parts can be delivered by a CDN-P2P, or by a P2P-P2P, configuration. Note that the number of peers in simulations is fixed, but the amount of peers may vary in real scenarios.

All experiments of this section are performed 30 times. The inter-arrival times of the requested software are generated following a Normal distribution, with mean is based on this time divided by the number of requests. Content demand is proportional to the number of requests considered for each scenario, and divided by the number of network nodes. The server request indicator and the parameter indicating whether a request can be routed is 1 for all locations. These parameters are available for adjustment from the current CDN providers considered in our study. Our simulations do not include dynamic network conditions, such as traffic, since providers receive enough payment per content delivery that acceptable performances for users is guaranteed.

7.2 | Results

This section presents simulation results performed on the proposed hybrid network, where values for PeerSim parameters are given in Table 5. At CDN nodes, we consider bandwidth and storage values as those offered by MaxCDN, since this provider offers the best QoS at a low cost. We also assume a fixed 400 Mbps bandwidth on P2P nodes with a 26 âĆň/month flat-rate. Furthermore, each P2P node has enough disc capacity to store the whole software container. We already have defined container sizes in section 6 when describing testing scenarios. On the other hand, 20 ms delays have been introduced into CDN-P2P and P2P-P2P communications. Finally, we have also considered in our simulations that the number of nodes and the number of user/requests are the same. The request percentage stored in a buffer is 50% of the whole capacity. This percentage value has been adjusted after performing 100 simulations by varying its value. This percentage value has been optmized after performing 100 simulations, where we vary this value. This minimum value enables the container deployment on response times that guarantee the system QoS.

We suppose a maximum deployment time of 2 min for software containerized applications between 5 GB and 10 GB. However, when the container size exceeds 10 GB, the deployment response time increases and surpasses 2 minutes, even if the whole container is delivered through the CDN. Thus, we estimate 3 minutes as an acceptable deployment time, which guarantees the network QoS. Figures 8 to 11 present our simulation results. In particular, they show that multinationals with a large number of employees could save between 5%-20%, when adopting a hybrid solution based on our proposed Cost-Efficient QoS-Aware model, rather than just renting a commercial CDN. In this case, savings are higher when container sizes do not exceed 10 GB.

On the other hand, savings are worthless in environments with few users. In particular, our hybrid infrastructure does not provide great benefits to SMEs with only few users (see Figures 10 and 11). In such cases, fixed costs, arising mainly from renting the broadband service, result comparable to deploying costs of the whole container through a CDN. Hence, the overall cost significantly increases due to fixed costs. Therefore, in these contexts our architecture does not bring important benefits.

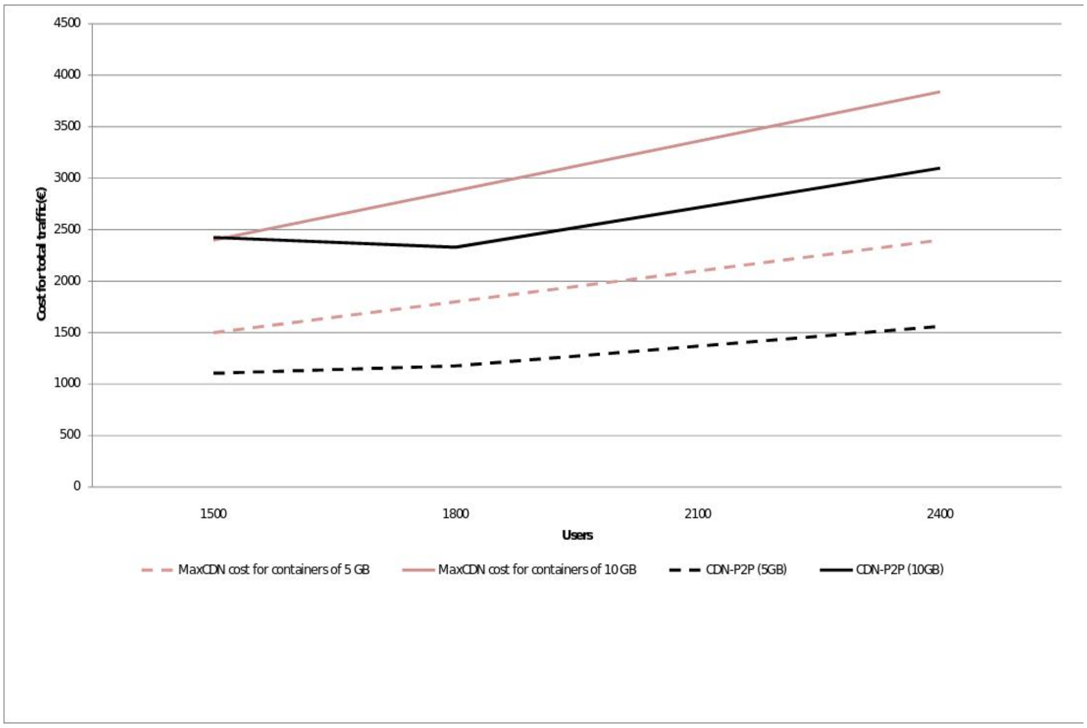

FIGURE 8 Test of our CDN-P2P architecture to deploy containers size of 5 GB and 10 GB for Scenario # 1

8 | CONCLUSIONS AND FUTURE WORK

This paper have presented a novel cost-efficient QoS-aware model to assess costs of containerized software distribution, and a hybrid CDN-P2P architecture to reduce such costs, making IT infrastructures more agile, in both consolidated corporations and emerging SMEs. We demonstrate that by combining both our architecture and model, companies could achieve higher economic savings, when compared to the case of renting a commercial CDN. A throughout analysis of existing commercial CDNs for software distribution has been conducted, concluding that MaxCDN is the one providing the lowest costs for a given QoS minimum level. To better understand the parametrization of these architecture and model, key metrics for software delivery are required, as opposed to some related works on hybrid networks that mainly focus on media streaming. To this end, a categorization of crucial features for software distribution has been presented, along with some performance metrics. This categorization provides suitable metrics that have been subsequently used to optimize relevant architecture and model parameters. Metrics depend upon content properties (e.g. content size, partial downloads, or attached/detached data), and also on user related parameters (e.g. distribution, number, or location of requests). The integer-programming simulation software PeerSim has been modified to test both the hybrid CDN-P2P architecture, and the novel cost-efficient QoS aware model, under various realistic scenarios. Testing results show the possibility of deploying over 20% of the whole container traffic through a P2P infrastructure, and this applies to both multinationals and SMEs. Our simulation results also reveal that a hybrid architecture is valuable when container sizes do not exceed 10 GB. It is important to remark that our architecture and model provide economic savings between 5% and 20%, with respect to just renting resources from a commercial CDN, and guarantee desired QoS in terms of response times

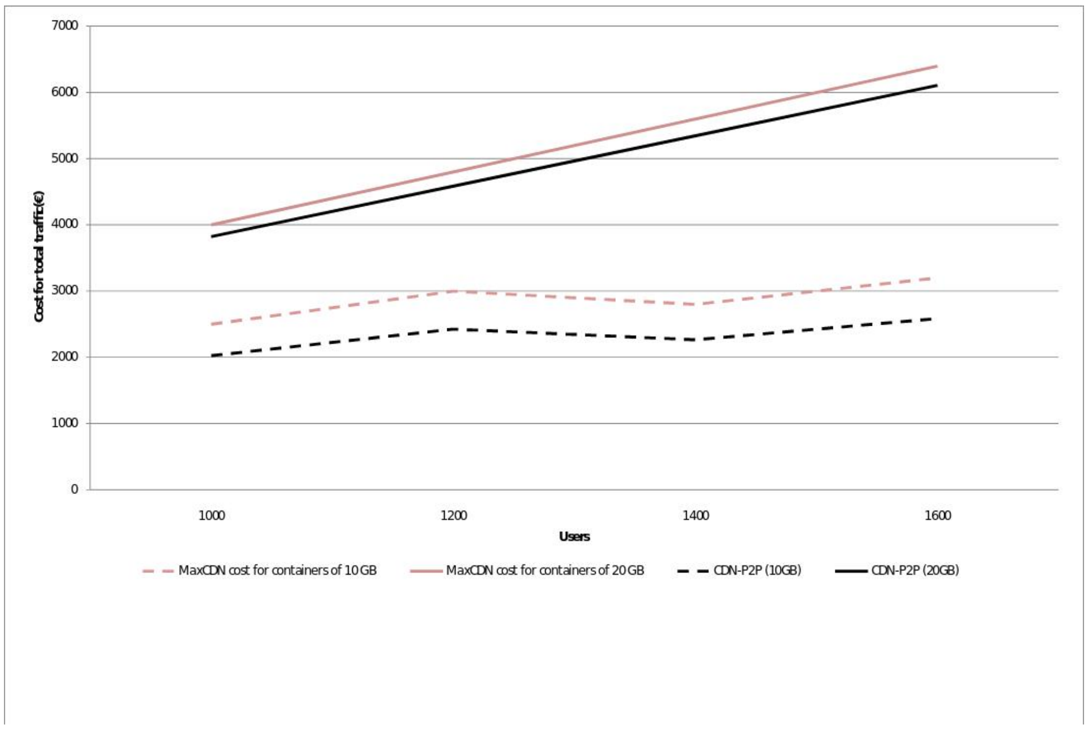

FIGURE 9 Test of our CDN-P2P architecture to deploy containers size of 10 GB and 20 GB for Scenario # 2

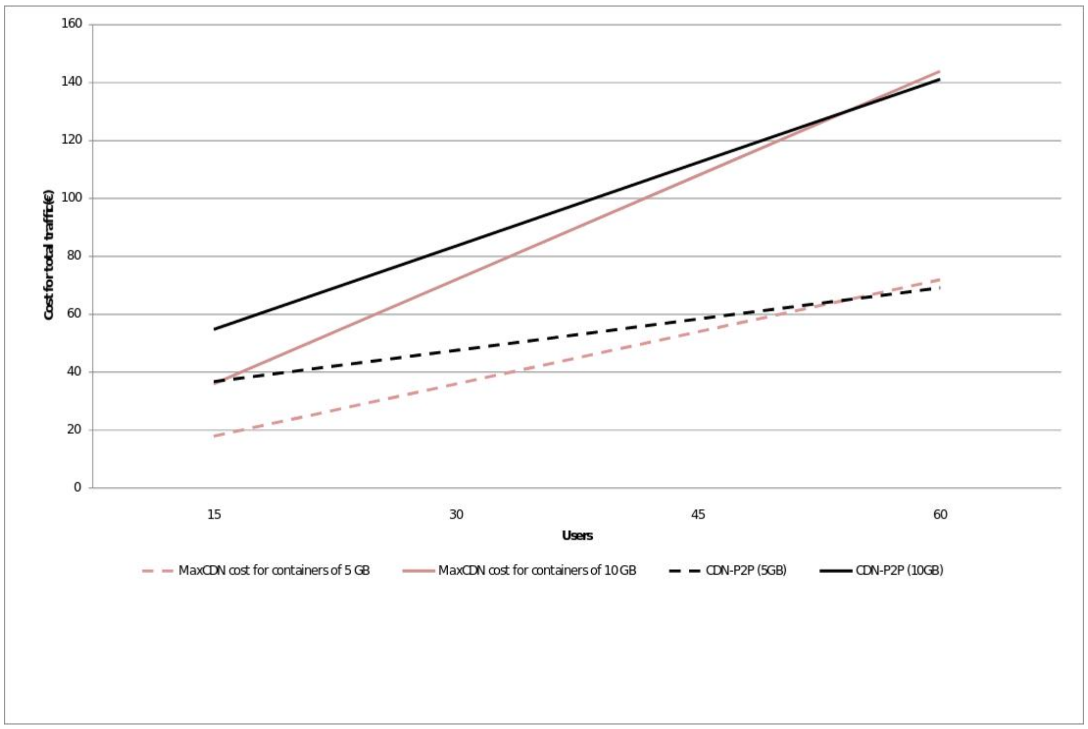

FIGURE 10 Test of our CDN-P2P architecture to deploy containers size of 5 GB and 10 GB for Scenario # 3

and the number of served requests. Future work should focus on the generalization of our simulator in order to consider a variable number of P2P nodes. We also consider that extending our current architecture and cost-efficient model to fog computing environments is a feasible and promising research topic.

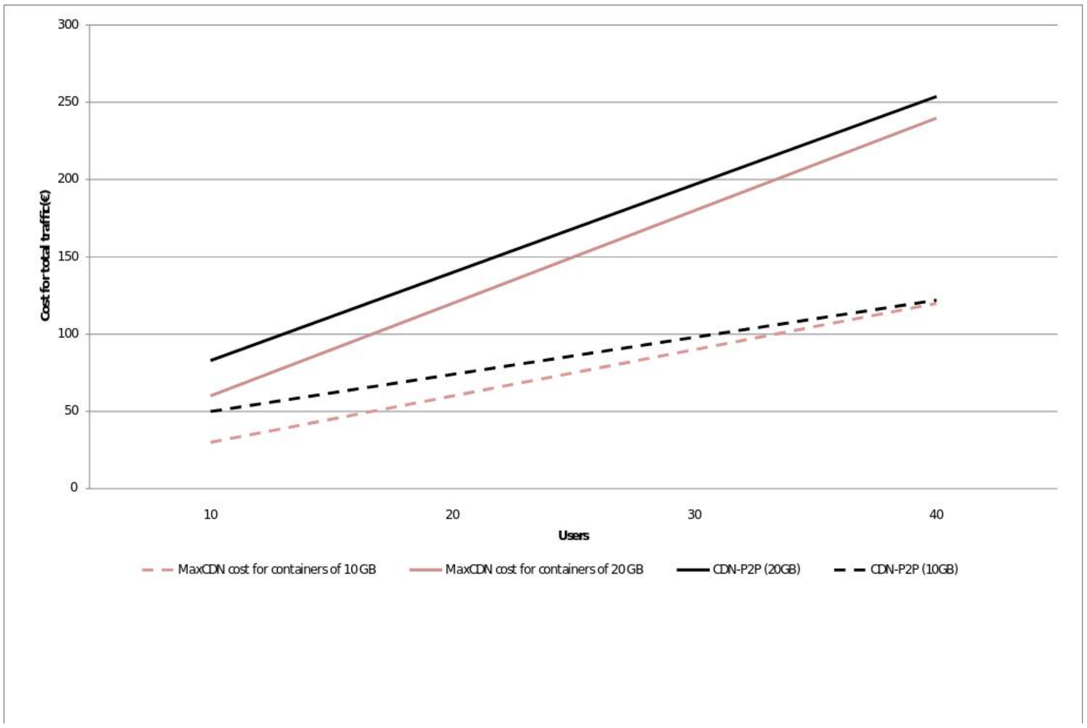

FIGURE 11 Test of our CDN-P2P architecture to deploy containers size of 10 GB and 20 GB for Scenario # 4

References

1. Da Silva L, Abreu F. Software Distribution to Remote Locations. In: 15th European Conference on Pattern Languages of Programs. ; July 7–11, 2010; Irsee, Germany.
2. Walters R. Bringing IT Out of the Shadows. *Network Security* 2013; 2013(4): 5–11.
3. Datadog . 8 Surprising Facts about real DOCKER adoption. https://www.datadoghq.com/docker-adoption/; . Accessed: 2019-10-19.
4. Portwox . 2017 Annual Container Adoption Survey: Huge Growth in Containers. https://portworx.com/2017-container-adoption-survey; . Accessed: 2019-10-19.
5. Lee H, Fox G. Efficient Software Defined Systems Using Common Core Components. In: IEEE 10th International Conference on Cloud Computing. ; June 25–30, 2017; Honolulu, Hawaii, United States: 407–414.
6. Johnston S. Introducing the Modernize Traditional Apps Program, in DockerCon. https://blog.docker.com/2017/04/modernizing-traditional-apps-with-docker/; . Accessed: 2019-10-19.
7. Li W, Kanso A. Comparing Containers versus Virtual Machines for Achieving High Availability. In: IEEE International Conference on Cloud Engineering. ; 2015; Tempe, AZ, USA.
8. Liu X, Dobrian F, Milner H, et al. A case for a coordinate internet video control plane. In: ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications. ; August 13–17, 2012; Helsinki, Finland: 359–370.
9. Wendell P, Freedman M, Milner H, et al. Going viral: flash crowds in an open cdn. In: 2011 ACM SIGCOMM Conference on Internet Measurement. ; November 02–04, 2011; Berlin, Germany: 549–558.
10. Liu H, Wang Y, Yang Y, Wang H, Tian C. Optimizing cost and performance for content multihoming. In: ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications. ; August 13–17, 2012; Helsinki, Finland: 371–382.

11. Cong X, Shuang K, Su S, Yang F. An efficient server bandwidth costs decreased mechanism towards mobile devices in cloud-assisted p2p-vod system. Peer-to-peer. *Peer-to-Peer Networking Applications* 2014; 7(2): 175–187.

12. Pathan A, Buyya R. A taxonomy and survey of content delivery networks. Tech. Rep. Technical Report, Grid Computing and Distributed Systems Laboratory; 2007.

13. Bronzino F, Gaeta R, Grangetto M, Pau G. An adaptive hybrid CDN/P2P solution for content delivery networks. In: Visual Communications and Image Processing. ; November 27–30, 2012; San Diego, CA, USA.

14. Lu Z, Wang Y, Yang Y. An analysis and comparison of CDN-P2P-hybrid content delivery system and model. *Journal of Communications* 2012; 7(3): 232–245.

15. Garmehi M, Analoui M, Pathan M, Buyya R. An economic replica placement mechanism for streaming content distribution in Hybrid CDN-P2P networks. *Computer Communications* 2014; 53: 60–70.

16. Zhang G, Liu W, Zou Y. Simulator for hybrid CDN-P2P video-on-demand systems. In: 9th International IEEE Conference on Communication Software and Networks. ; May 6–8, 2017; Guangzhou, China.

17. Ha T, Kim J, Nam J. Design and Deployment of Low-Delay Hybrid CDNâĂŞP2P Architecture for Live Video Streaming Over the Web. *Wireless Personal Communications* 2017; 94(3): 513–525.

18. Jia Q, Xie R, Huang T, Liu J, Liu Y. The collaboration for content delivery and network Infrastructures: A survey. *IEEE Access* 2017; 5: 18088–18106.

19. Anjum N, Karamshuk D, Shikh-Bahaei M, Sastry N. Survey on peer-assisted content delivery networks. *Computer Networks* 2017; 116: 79–95.

20. Michael C. Do It Yourself CDN: Netflix, Comcast and Facebook, Bizety. https://www.bizety.com/2016/02/23/do-it-yourself-cdn-netflix-comcast-and-facebook/; . Accessed: 2019-10-19.

21. Yeadon N, Garcia F, Hutchison D, Shepherd D. Filters: QoS support mechanisms for multipeer communications. *IEEE Journal on Selected Areas in Communications* 1996; 14: 1245–1262.

22. Evensen K, Kaspar D, Griwodz C, Halvorsen P, Hansen A, Engelstad P. Improving the Performance of Quality-adaptive Video Streaming over Multiple Heterogeneous Access Networks. In: ; February 23–25, 2011: 57–68.

23. Trestian R, Comsa IS, Tuysuz MF. Seamless Multimedia Delivery within a Heterogeneous Wireless Networks Environment: Are we there yet?. *IEEE Communications Surveys & Tutorials PP(99)* 2018; 20: 945–977. doi: 10.1109/COMST.2018.2789722

24. Gkantsidis C, Karagiannis T, Rodriguez P, Vojnović M. Planet scale software updates. In: SIGCOMM'06. ; September 11–15, 2006; Pisa, Italy: 11–15.

25. Passarella A. A Survey on Content-centric Technologies for the Current Internet: CDN and P2P Solutions. *Computer Communications* 2012; 35(1): 1–32.

26. Yin H, Liu X, Zhan T, et al. Design and deployment of a hybric CDN-P2P system for live video streaming: experiences with LiveSky. In: 17th ACM International Conference on Multimedia. ; October 19–24, 2009; Beijing, China: 25–34.

27. Xu D, Kulkarni S, Rosenberg C, Chai HK. Analysis of a CDN-P2Phybrid architecture for cost-effective streaming media distribution. *Multimedia Systems* 2006; 11(4): 383–399.

28. Hu M, Luo J, Wang Y, Veeravalli B. Practical Resource Provisioning and Caching with Dynamic Resilience for Cloud-Based Content Distribution Networks. *IEEE Transactions on Parallel and Distributed Systems* 2014; 25(8): 2169–2179.

29. Sahoo J, Glitho R. Greedy heuristic for replica server placement in Cloud based Content Delivery Networks. In: IEEE Symposium on Computers and Communication. ; June 27–30, 2016; Messina, Italy: 302–309.

30. Krishnan P, Raz D, Shavitt Y. The cache location problem. *IEEE/ACM Transactions on Networking* 2000; 8(5): 568–582.

31. Kalpakis K, Dasgupta K, Wolfson O. Optimal Placement of Replicas in Trees with Read, Write, and Storage Costs. *IEEE Transactions on Parallel and Distributed Systems* 2001; 12(6): 628–637.

32. Kubernetes . Horizontal pod Autoscaling. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/; . Accessed: 2019-10-19.

33. Poddar R, Vishnoi A, Mann V. HAVEN: Holistic Load Balancing and Auto Scaling in the Cloud. In: 7th International Conference on Communication Systems and Networks. ; January 6–10, 2015; Bangalore, India.

34. Abranches C. dM, Solis P. An Algorithm Based on Response Time and Traffic Demands to Scale Containers on a Cloud Computing System. In: IEEE 14th International Symposium on Network Computing and Applications. ; September 28–30, 2015; Cambridge, MA, USA: 343–350.

35. Overview C. CDN resource website. https://www.cdnoverview.com/; . Accessed: 2019-10-19.

36. Vakali A, Pallis G. Content Delivery Networks: Status and Trends. *IEEE Internet Computing* 2003; 7(6): 68–74.

37. Microsoft . Docker Hub. https://hub.docker.com/u/microsoft/; . Accessed: 2019-10-19.

38. Harter T, Salmon B, Liu R, Arpaci-Dusseau A, Arpaci-Dusseau R. Slacker: Fast Distribution with Lazy Docker Containers. In: 14th USENIX Conference on File and Storage Technologies. ; February 22–26, 2016; Santa Clara, CA, USA: 181–195.

39. KeyCDN . keyCDN Website. https://www.keycdn.com/; . Accessed: 2019-10-19.

40. MaxCDN . MaxCDN Website. https://www.maxcdn.com/; . Accessed: 2019-10-19.

41. CDN77 . CDN77 Website. http://www.cdn77.com; . Accessed: 2019-10-19.

42. BunnyCDN . BunnyCDN Website. https://bunnycdn.com/; . Accessed: 2019-10-19.

43. James H. 25 Best CDN Providers 2019. https://haydenjames.io/best-cdn-providers/; . Accessed: 2019-10-19.

44. Hoa D, Silverton T, Fourmaux O. A novel Hybrid CDN-P2P mechanism for effective real-time media streaming. In: Semantic scholar. : 1–8.

45. Commission E. Broadband market developments in the EU 2017. https://ec.europa.eu/digital-single-market/en/european-digital-progress-report; . Accessed: 2019-10-19.

46. Commission E. Connectivity for a European Gigabit Society. https://ec.europa.eu/digital-single-market/en/policies/improving-connectivity-and-access; . Accessed: 2019-10-19.

47. Montresor A, Jelasity M. PeerSim: a scalable P2P simulator. In: 9th International Conference on Peer-to-Peer (P2P'09). ; September 8–11, 2009; Seattle, WA, USA: 99–100.

□