CONVINcE
D3.3.2

Design of energy efficient CDN (including data centers and cloud)

Editor: Saeed Bastani (LU)

Reviewer: Reza Farahbakhsh, Samin Mohammadi (IMT)
Authors: Saeed Bastani (LU), Yannick Carlinet, Yassine Naghmouchi, Nancy Perrot (OL), Arto Heikkinen (UO), Reza Farahbakhsh, Samin Mohammadi, Noel Crespi (IMT), Miika Komu (ER)

Executive Summary

This deliverable is focused on exploring new architectures, techniques, and solutions that can facilitate the saving of energy consumed for content distribution, while, in the meantime, the required Quality of Service (QoS) constraints are satisfied. Of many possible ways of saving energy, we have addressed a minimal set of key mechanisms that can lead to substantial energy saving and improved QoS satisfaction when processing and delivering contents to the end-users. A brief description of such mechanisms addressed in this report is presented as follows:

1. We propose an end-to-end approach to describe the energy usage of video delivery within a content delivery framework, and use this to investigate the energy usage behavior of two popular coding schemes, namely, H.264/AVC and H.265/HEVC. Our study based on the proposed model is backed up by measurements of encoding and decoding energy usage of a sample video and shows that, from an end-to-end perspective, taking into account all the elements of a content delivery network, neither of the coding formats is always dominant in terms of energy saving. We also find that the popularity of video content is a key parameter for predicting which encoding scheme saves most energy. In particular, we find that H.265 encoded content results in lower energy usage if the content is highly popular. On the other hand, for a content with predicted low popularity, more saving is achieved if H.264/AVC is used. This lead us to calculate a hybrid content delivery scheme, where the contents with low popularity are encoded and delivered in H.264/AVC format, whereas content of high popularity are encoded and delivered in the H.265/HEVC format.
2. User Generated Content (UGC) is projected to make up a significant part of the total Internet traffic in the future. As such, it will significantly contribute to the total cost for Internet traffic worldwide. Arguably, UGC is a suitable content type for which the Internet Service Providers (ISPs) can take initiative and enter the content delivery market. However, despite its significance, UGC content management has attracted very little research attention, and the existing works stop short of developing placement and delivery solutions for UGC. Hence, we are motivated to address this content type, and exploit its properties to support ISPs in making optimal placement decisions. Specifically, we leverage the inherent tie between UGC and social networking context, take into consideration the persistence limitation of UGC (in contrast to commercial content), and derive model with the objective to minimize power usage. Also, derived from the problem formulation, we propose an online algorithm which enables each ISP to individually decide which contents should be placed and served locally. We provide simulation results showing that the proposed algorithm performs close to optimal in terms of power used for content delivery.
3. When load balancing data centres with all servers active to serve jobs, the result can be excessive energy usage. On the other hand, using dynamic server provisioning, the number of servers that serve requests can be reduced by turning off idle servers and thereby save energy. However, such a scheme, usually increases the risk of instability of server queues. In this work, we analyze the trade-off between energy usage and stability of servers in a data center when we balance the load by dispatching arriving jobs. We propose an algorithm, termed SEOL, to solve a stability and energy objective stochastic optimization problem with a high degree of flexibility to handle the trade-off between these two objectives. We show the performance of our proposed algorithm in minimizing the risk of queue length growth as well as the number of active servers needed to serve jobs, and compare it with a number of well-established commercial load balancing mechanisms.
4. We conduct a fundamental study on cost (or utility) sharing among multiple ISPs participating in content distribution reveals that efficient mechanisms exist to reach a trade-off between the total cost saved and the share of resources (e.g. bandwidth) allocated to individual ISPs. In particular, we showed that bargaining solutions inspired from game theory have a desirable potential to achieve such a trade-off. In addition, these methods are simple and parsimonious, making them easy to implement as a third party component responsible for resource sharing between ISPs. This approach can be employed in the IETF CDN Interworking (CDNI) framework to support efficient interworking among multiple (Telco) CDNs.
5. Cooperative content management plays a key role in QoS satisfaction and reducing the overall energy consumption of content delivery networks. In this document, we have addressed this solution by developing an optimization problem, and proposed a number of cooperative content

caching and distribution algorithms. Our results verify that the proposed cooperative algorithms behave very close to optimal in terms of content access latency, while in the meantime achieve a substantial saving of power consumed for content transfer.

6. In-network caching coupled with equipment power-off is another key solution to energy saving in the next generation content distribution networks. It has been shown that this energy-aware caching technique is superior to the conventional shortest path-based caching. In view of this, we address the problem of energy saving in content oriented networks, which explores the optimal subset of caches and links that could be turned off to minimize energy while finding a feasible routing in the network, which may not be necessary on a shortest path under capacity constraints. We address this problem within a generic case of object caching and traffic routing problem in arbitrary graph-based network topologies. We have investigated this problem by developing an optimization framework, which is then validated by solving instances of real network topologies. Based on several network performance metrics, we have shown the impacts and the gains of introducing energy-aware on a real telecommunication network. Furthermore, we have proposed a heuristic method, allowing us to show the benefits of our model compared to the classical routing on shortest path-based caching model.
7. Resource-awareness is a key factor when involving the end-user devices (e.g. mobile phones) in content caching and delivery process. This coupled with the proliferation of versatile client browsers offer the advantages of flexibility and ease of implementation. In light of this, we present mechanisms for optimizing energy consumption in video delivery to mobile devices. Our focus is on distributed mechanisms for Web-based video delivery, as the emerging WebRTC technology allows direct communication between modern web browsers for streaming video and transmitting arbitrary data. We present a design of resource-aware distributed content delivery based on WebRTC, which enables building scalable and energy-efficient video delivery systems for mobile Web.
8. We propose to use social information of users extracted from the context of social networks coupled with the opportunistic use of available resources in the users' devices for content caching and sharing. We believe that the this solution will offer a high level of self-scalability, similar to the general case of Peer-to-Peer (P2P) content networking paradigm.
9. Our contribution to energy-efficient cloud networking design is built upon the current trend in implementing complex software, where the system is split up into several independent components — micro-services. In video processing applications, the service typically takes the form of a chain, where the video stream is processed by a component at a time. This allows controlling for each component where the component runs. Because of the low overhead of containers, it is feasible to run a dedicated chain of containers for each stream. This allows controlling the processing of each stream with a fine granularity and to include only the processing components required for the particular stream. Taking this into consideration, we propose a solution and identify each device joining the network and set up a chain of containers for the particular device. Our solution is based on an orchestrator and a SDN controller. We have done a prototype implementation of this approach to study the performance and power consumption.
10. In Software Defined Networking (SDN), the service function chaining technique augmented with minimal edges and nodes is thought as another primary solution to achieve a substantial reduction of overall power consumption. In our work, we address the problem of optimal deployment of service function chains in terms of maximal energy savings on a network by computing paths and function installation altogether. To this end, we have developed an optimization framework to solve the service chaining problem with minimal active resources in terms of edges and nodes. To cope with the time complexity of obtaining the optimal solution of this problem, we have proposed lightweight heuristic algorithms and studied their performance. We further investigate this problem by exploring exact procedure(s) to obtain high performance solutions with controlled time complexity.
11. Controller placement in SDN is also regarded as a key technique where energy saving can be secured. This can be achieved if the number of required controllers is minimized. We address this case thoroughly by exploring controller placement solutions that yield the minimal set of controllers, the optimal placement of the controllers in the network, and the set of assigned nodes to each controller.
12. Load balancing has proven as a key technique to the reduction of power consumption. Load balancing in an SDN-based network of Data Centers (DCs), especially in the case of a video service, is a particularly interesting problem. SDN is essential here because it permits a global

approach. Also, it allows for remote management of the virtual machines responsible for processing the user's requests. We have addressed the problem of load balancing within this context, and presented our preliminary results. We observe that the gains of sleep mode are 38% and 31% in the cases of large DCs and fog computing, respectively. However, the load balancing mechanism leads to a gain of 11% in the case of large DCs, and an astounding 69% in the case of fog computing. These results, however, are obtained with the optimal algorithm that knows *a priori* the future requests so as to perform a perfect request assignment. Such an algorithm cannot be implemented in reality, but these results show that a well-designed load balancing algorithm can achieve a significant benefit, in terms of reducing the energy consumption. The design of such algorithm(s) will be addressed and presented in this final design document.

Table of Contents

Table of Figures

Table of Tables

DOCUMENT HISTORY AND ABBREVIATIONS

Document history

Version	Date	Description of the modifications
0.1	2017-04-01	Table of Contents (ToC)
0.2	2017-07-05	Contributions from all partners are integrated
0.3	2017-07-17	Reviewers comments are applied
1.0	2017-07-20	Final version is prepared

Abbreviations

BGP	Border Gateway Protocol
CCN	Content-Centric Networking
CDN	Content Delivery Network
DC	Data Center
DP	Dynamic Provisioning
EON	European Optical Network
HEVC	High Efficiency Video Coding
IP	Internet Protocol
ISP	Internet Service Provider
MPEG	Moving Picture Experts Group
NFV	Network Functions Virtualization
OLT	Optical Line Termination
ONU	Optical Network Unit
OSN	Online Social Networking
P2P	Peer to Peer
QoE	Quality of Experience
QoS	Quality of Service
RM	Resource Management
SDN	Software Defined Networking
SLA	Service Level Agreement
SVC	Scalable Video Coding
ToD	Time of Day
UGC	User Generated Content
VM	Virtual Machine
VoD	Video on Demand
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing

1 INTRODUCTION

The Internet, nowadays, has witnessed a tremendous growth in the volume of contents released and distributed worldwide. This phenomenon is partly caused by the proliferation of versatile user devices, wireless technologies and, above all, the explosion of high volume video contents published by enterprises and also by end-users. Notably, a great extent of this high volume traffic, such as video streaming, poses stringent Quality of Service (QoS) requirements, which makes the task of content delivery even more challenging. These phenomena, hand-in-hand, have given rise to a massive deployment of dedicated content distribution architectures and technologies.

Content Delivery Networks (CDNs), alone, account for around 80% of content distribution worldwide. Naturally, the capital and operational costs of these large scale networks also grow with the volume of traffic they handle. Unarguably, energy consumption, as the present-time concern in the ICT sector, also grows with the amount of traffic exchange. It is therefore a vital task to explore new techniques and solutions that, on one side, can be easily incorporated in the existing and the future-generation content deliveries networks, and, on the other side, reduce the energy consumed for massive scale content transfer. Along this line, optimal content management and resource-aware solutions to content delivery are embraced as sustainable mechanisms for efficient utilization of the available resources (including the centralized equipment and end-user devices) while, in the meantime, the QoS requirements are not compromised.

Cloud networking is also playing a vital role in the processing and delivery of Internet contents, since the delivery of content, particularly video contents, involves a significant amount of processing such as transcoding before delivering the content to the end-users. Therefore, improvements of the enabling functionalities of cloud networks such as media processing and virtualization can be seen as another important step towards cost saving in the overall process of content delivery.

The last, but not the least, is to employ the trending techniques in the context of Software Defined Networking (SDN) which offer a significantly high level of flexibility in dealing with the control plane of data communication networks. In view of this, one can see the integral elements of SDNs such as service and network function chaining as the key tools for facilitating energy efficient content transfer in the various layers of the underlying carrier networks, i.e. core, access, and aggregation.

In this document, we focus on the CDNs, SDNs, and cloud networking and propose new solutions for energy-efficient content processing/transfer. A common design approach that we aim to follow throughout this final design is to treat energy saving and QoS satisfaction as a coupled design goal, and to seek trade-offs if they conflict in one way or another.

1.1 Scope

The CONVINcE project has promised to address the challenge of reducing the power consumption in IP-based video networks with an end-to-end approach, from the Head-End where contents are encoded and streamed to the terminals where they are consumed, taking into consideration the content delivery networks and the core and access networks.

Power saving in the network, as targeted by CONVINcE WP3, will focus on the design and the development of solutions to enable energy-efficient delivery of video streams in all parts of the network. The main objectives of this work package are to study and develop consolidated solutions to reduce energy consumption in the network nodes (core/access/aggregate) and in the content processing and delivery overlays including CDNs and clouds.

In the WP3, the task T3.3 is particularly dedicated to explore new mechanisms for energy efficient content distribution, taking into account the QoS requirements. The major architectural and technical solutions to content distribution consist of CDNs, cloud networking and virtualization, and Software Defined Networking (SDN), with the latter regarded as a flexible technique applied to CDNs, clouds, and the more general case of traffic routing.

This document describes the Deliverable D3.3.2 “Design of energy efficient CDN (including data centres and cloud)” as part of task T3.3 within the WP3 “Power saving in the network” of the CONVINcE project.

The scope of this deliverable is as follows:

- CDNs and data centers
- Cloud Networking
- SDN and Network Functions Virtualization (NFV)

While there are many possible ways of enhancing energy consumption of content delivery process, our design of energy efficient CDN and clouds, as the subject of this deliverable, is centered on key mechanisms where maximum saving can be sought after. Accordingly, in this design document, we address the following key components, and present our proposed solutions and preliminary results:

- We regard content management as a key element in content distribution process, and propose efficient content management solutions for overlay CDNs, interworking CDNs as well as in-network caching in the future-generation Content-Centric Networks (CCNs).
- We propose a novel idea to content management, which relies on the social information of users and the resources contributed by user devices. This approach resembles the Peer-to-Peer (P2P) networking paradigm, and thus, achieves a similar property of self-scalability.
- We address resource-aware content distribution, and propose a distributed mechanism for content transfer among mobile nodes using a hybrid architecture. This is envisioned to enable the possibility of content access from peer devices or from a centralized entity, e.g. a centralized content server.
- We propose a novel mechanism for energy-efficient cloud networking based on container technology, implement this solution and show its performance.
- Concerning SDN, our design addresses novel solutions to service function chaining in SDN, controller placement and geographical load balancing.

This deliverable completes the previous deliverable D3.3.1 (Initial design of energy efficient CDN and cloud) in several directions as follows: i) an extensive survey of CDN evaluation measures and CDN energy consumption (Sections 2.2 and 2.3), ii) a finalized and peer-reviewed study on the anatomy of end-to-end energy consumption in CDN with regards to different video coding schemes (Section 3.1), iii) an online energy-efficient content placement mechanism for User Generated Content (UGC) (Section 3.3), iv) a trade-off mechanism for server stability and energy usage in data centers (Section 3.4), v) a game-theoretic cost/utility sharing among multiple TelcoCDNs (Section 3.5), vi) energy-efficient cloud networking (Section 4), and vii) completed work on geographical load balancing in SDN-based data centers. (Section 5.2.3).

1.2 Document structure

This document is organised as follows:

- Section 2 presents the state-of-the-art in CDN design, CDN performance measures, energy consumption statistics in CDN, the relation between social networking and CDN, and the existing solutions to CDN performance improvements.
- Section 3 presents the solutions proposed by CONVINcE project partners towards efficient content management and delivery in CDNs.
- Section 4 describes the state-of-the-art in energy-efficient cloud networking design, and a solution proposed by CONVINcE project partners.
- Section 5 presents the state-of-the-art of SDN, and the solutions proposed by CONVINcE project partners to the key elements of SDN including service function chaining, controller placement and load balancing.

- Section 6 presents the conclusion remarks and the highlights of this deliverable.

2 CONTENT DELIVERY NETWORKS (CDN)

This section is organized as follows. We present an overview of the state-of-the-art of energy-efficiency mechanisms in CDNs and data centres (DCs) in Section 2.1, followed by an in depth survey of CDN performance measures (Section 2.2), analysis of CDN energy usage (Section 2.3), CDN performance improvement techniques proposed in the literature (Section 2.4), and the interplay between social networking and CDN (Section 2.5).

2.1 State-Of-The-Art

Energy efficient content delivery has drawn a significant research attraction recently. These include both empirical and theoretical works and cover a diverse range of topics from energy efficient content management to energy-aware load balancing and dynamic provisioning of servers (or server clusters) and network elements (i.e. links and routers/switches). Yet, other works conducted comparative studies on the energy efficiency of various content delivery architectures such as next-generation content-centric networks (CCNs) and the conventional overlay content delivery networks (i.e. conventional CDNs). In the following, we present a number of leading research works and highlight the implications of these studies from the perspective of energy-efficient design of content distribution networks and data centres. More detailed complementary background and related work are presented throughout the individual sections of this document.

Dynamic provisioning of servers and network elements has been regarded as the major energy saving mechanism in data centres and in the CDNs comprising multiple distributed DCs. In the following, a number of studies concentrated on Dynamic Provisioning (DP) techniques are described. A DP technique is usually accompanied by a suitable resource management scheme (RM) which refers to the allocation of user demands to the servers (or network elements), taking into account the on/off status of servers/network elements decided by the exploited DP mechanism. Xu et. al. [1] developed a theoretical framework and proposed two strategies with different objectives. The first strategy was to enhance cache hit ratio via pooling and sharing caches among servers in a cluster. In a second strategy, they targeted power saving by consolidating user requests and routing them to fewer servers while putting unused servers in sleep mode. Chiaraviglio et. al. [2] proposed GreenCoop as a solution to the problem of jointly minimizing energy consumption of Internet Service Provider (ISP) and CDN operator. Figure 1 demonstrates GreenCoop framework comprising an ISP with four Point of Presence (PoP) and a CDN with two servers. In the figure, the density of users around each PoP represents the amount of local demands. GreenCoop has two different behaviours in peak and off-peak hours. In peak hours, as shown in Figure 1 (a), both servers are provisioned but the ISP temporarily deactivates its unused links and network elements after performing request resolution and routing to the servers. In off-peak hours, as shown in Figure 1 (b), a second type of energy saving mechanism can be enforced, which involves the consolidation of requests and routing them to only one of the CDN servers, and putting the other server in the sleep mode. Chiaraviglio et. al. [2] showed that GreenCoop can achieve up to 70% of energy saving. A positive aspect of GreenCoop lies in its ability to take into account end-to-end delay constraint and also the capacity constraints of servers and links. A limitation of GreenCoop is the degeneration of CDN DCs to contain only one server which is not a realistic assumption. Another limitation is the assumption of full information accessible to each player (ISP and CDN) about the other player which is, again, an unrealistic assumption. Similar studies employing DP-based power saving approach can be found in [3] [4] [5]. Like GreenCoop, these studies take into consideration some forms of QoS constraints while optimizing power consumption. However, most of these studies oversimplify the DCs by representing each DC with a single server. Only few works including [6] address the case of large scale DCs with multiple servers organized in clusters. In the solution proposed in [6], two modes of dynamic provisioning are supported: in a basic level, the requests are consolidated in the internal cluster within a DC in order to put some servers of the DC in idle mode. In a second mode, the requests are consolidated and routed to fewer DCs in order to shut down an entire DC.

Pushing the contents further to the network edge, i.e. beyond what is reachable by the conventional CDNs, has been addressed in recent research studies. In light of this, Valancius et. al. [7] proposed the idea of Nano Data centres (NaDa) for energy efficient content distribution. In contrast to the conventional data centres, NaDa caches the contents on the end-users' home gateways owned by

users and managed by the ISP (see Figure 2). Besides to the manageability of the content distribution framework, NaDa also provides scalability through the resultant mini-data centre environment.
NaDa achieves energy saving by reducing the traffic volume traversing the Internet, and by reducing the need for large scale content servers and network resources. As shown in Figure 3 (a) and Figure 3 (b), corresponding to the diurnal traces of YouTube and Netflix respectively, NaDa retains a significant gain of energy saving compared to the conventional data centre solution. Interestingly, with the traffic demand increasing (corresponding to peak traffic periods), the energy consumption gain in NaDa improves, too, reaching to about 30% saving compared to the conventional data centre approach. While NaDa promises significant energy saving, its capability to satisfy QoS is questionable due to its relying on low capability (processing) devices with intermittent availability (e.g. turning to sleep mode). Also, it is not clear if the users will have sufficient incentives to allow their uplink bandwidth be used by peers.

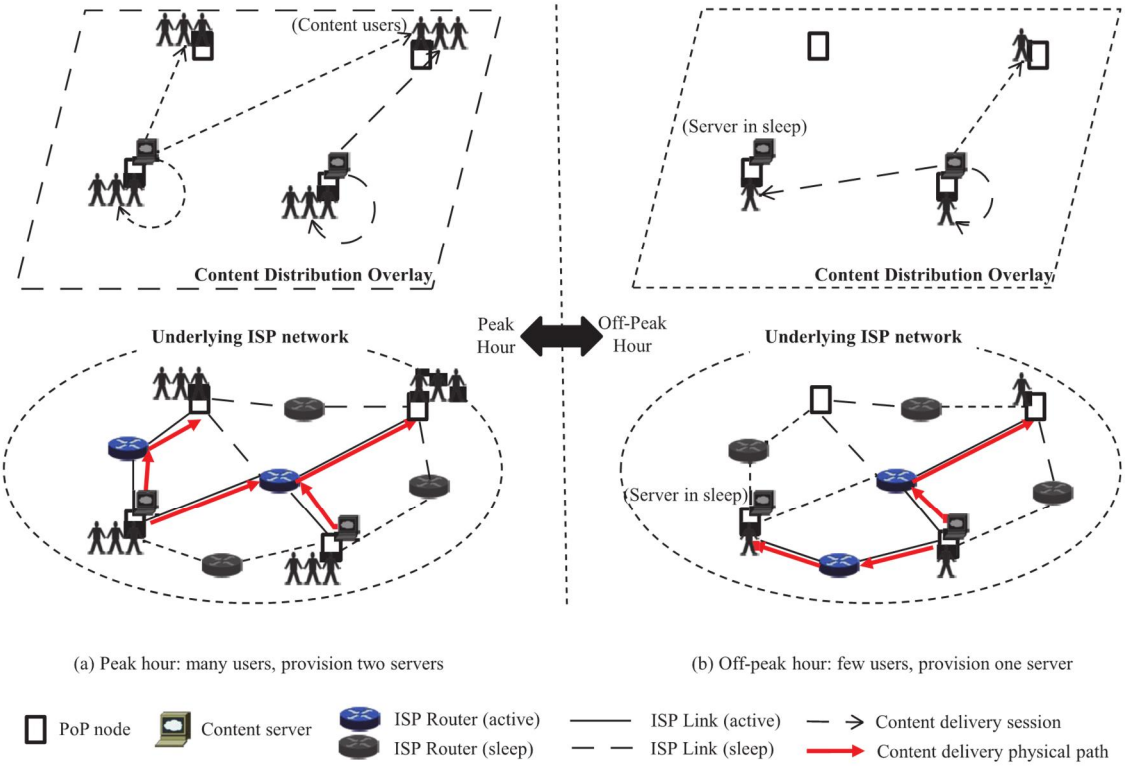


Figure 1: Operation diagram of GreenCoop [2]

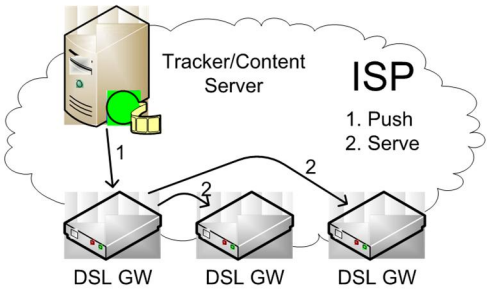


Figure 2: Nano Data centre architecture [7]

In-network caching, as a step towards full-featured content-centric networking (CCN), has attracted attentions as a means of energy efficient content distribution. Lee et. al. [8] proposed the idea of using CCN to achieve an energy efficient content delivery. In contrast with the conventional overlay CDNs where content caching is performed by dedicated (surrogate) servers, in-network caching fully relies on network elements (i.e. routers) to perform caching and content retrieval. The principal behind the energy efficiency of CCN is that the core network devices (e.g. core routers) are significantly more energy efficient compared to edge devices, home gateways (as in NaDa), and servers [9]. Indeed,

using trace-based simulations, Lee *et. al.* [8] showed that the CCN paradigm of content distribution is, in general, more energy efficient than both the conventional overlay CDNs and the more recent nano data centre approach. More specifically, they showed that, in order to gain the highest energy saving in CCN, the contents with high popularity should be cached and served in the edge routers whereas non-popular contents should be cached in the core network and closer to the original content servers. An in-depth comparative study of the competitive content distribution architectures has been conducted by Guan et. al. [10]. In this study, two different CDN architectures along with CCN architecture, as depicted in Figure 4, are compared from the perspective of energy efficiency. The CDN architecture in Figure 4 (a) represents the conventional case of present time content delivery networks such as Akamai, while the CDN architecture in Figure 4 (c) is a centralized architecture implemented in a single big data centre. In the latter, it is assumed that dynamic optical bypass is enabled.

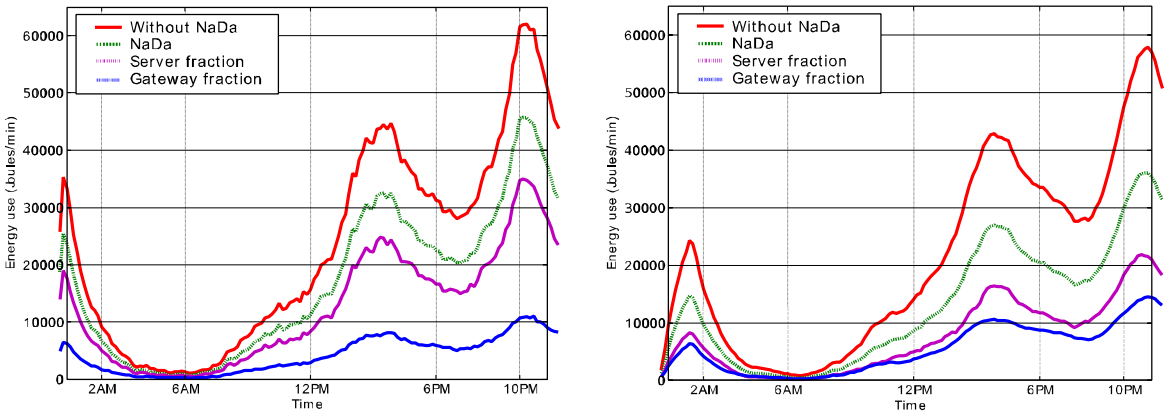


Figure 3: NaDa diurnal patterns of energy usage

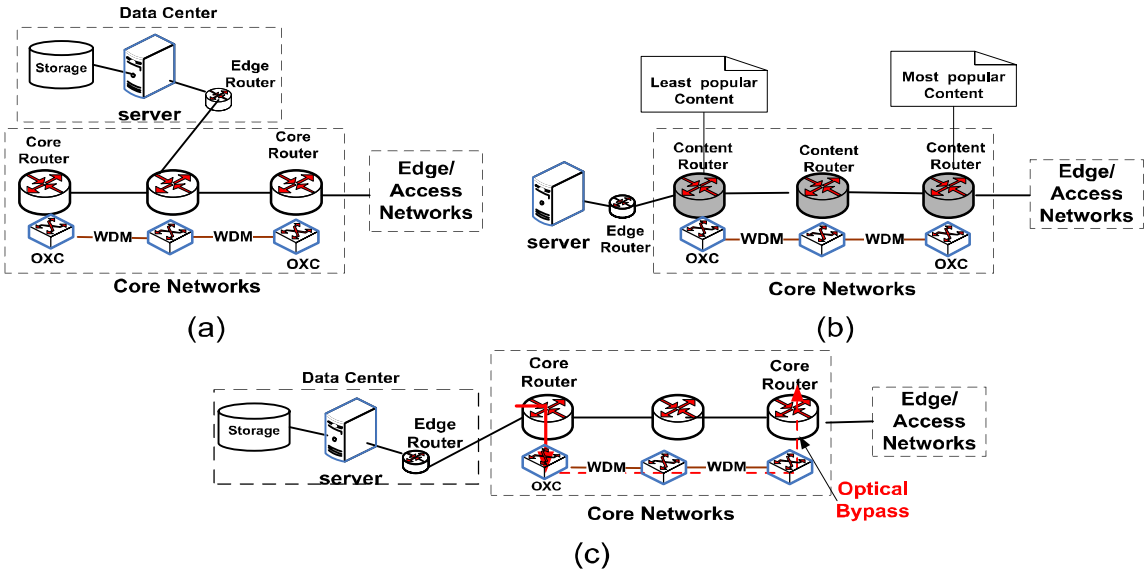


Figure 4: Content delivery architectures: (a) decentralized server-based CDN, (b) content-centric network (CCN), (c) centralized server-based CDN using dynamic optical bypass [10]

In their numerical study, Guan et. al. [10] used four different network topologies including a ring network with 64 nodes, an 8×8 grid with 64 nodes, an IP backbone with 24 nodes and European Optical Network (EON) with 19 nodes. The topologies are chosen in a way that the degree of connectivity (node degree) decreases with the order of topologies, i.e. minimum connectivity pertaining to the ring and the maximum connectivity belonging to EON topology. Figure 5 shows the energy consumption of CCN and CDN architecture with optical bypass enabled. It can be observed that for smaller demand rates the CDN with bypass outperforms the CCN, but for larger demand rates the

CCN gains better energy efficiency. Note that with the network connectivity increasing, the superiority of CCN takes place at smaller demand rates. Figure 6 demonstrates the energy consumption ratio between CCN and the conventional CDN architecture as a function of popularity exponent (the value of β in the Zipf's distribution adopted in their study) and the total number of contents or catalogue size (F). It can be observed that CCN outperforms the CDN for small catalogue sizes whereas CDN performs better with large catalogue sizes. Another observation is that with β increasing (corresponding to heavy tail Zipf distribution), the ratio of energy consumption also decreases, though not very significantly. Finally, the impact of topology, characterized by the number of nodes N and connectivity degree, on the efficiency ratio is evident from Figure 6. Network topologies with smaller N and stronger connectivity exhibit higher efficiency in CCN as compared to the conventional CDN.

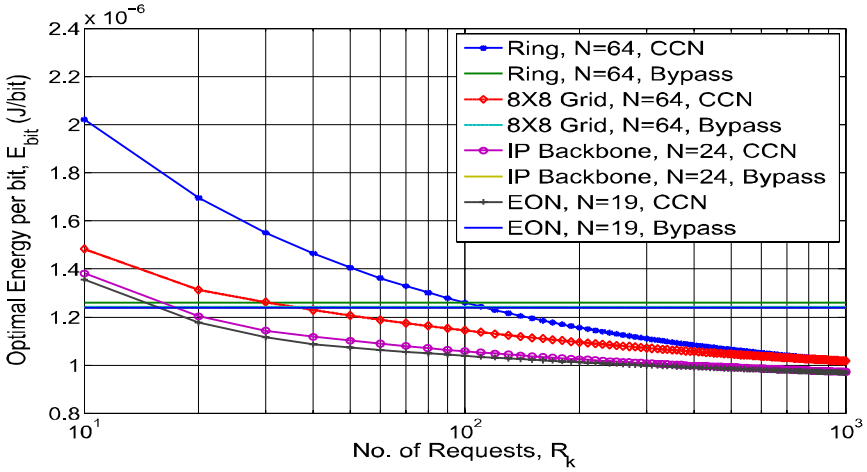


Figure 5: comparison of CCN and CDN with bypass, as a function of request rate

There are few other studies that investigated the energy efficiency of CCN architecture. Choi *et. al.* [11] studied the impact of cache location and also the cache capacity of content routers on the energy consumption behaviour of CCN. They developed two optimization problems with the objective of energy consumption minimization. The first problem was formulated for optimal cache placement in the routers, while the second problem describes the optimal cache volume to be available in each content router. It was shown that, in order to gain the benefits of CCNs, it is necessary to optimally place the caches, given that sufficient caching capacity and energy-proportional content routers are available.

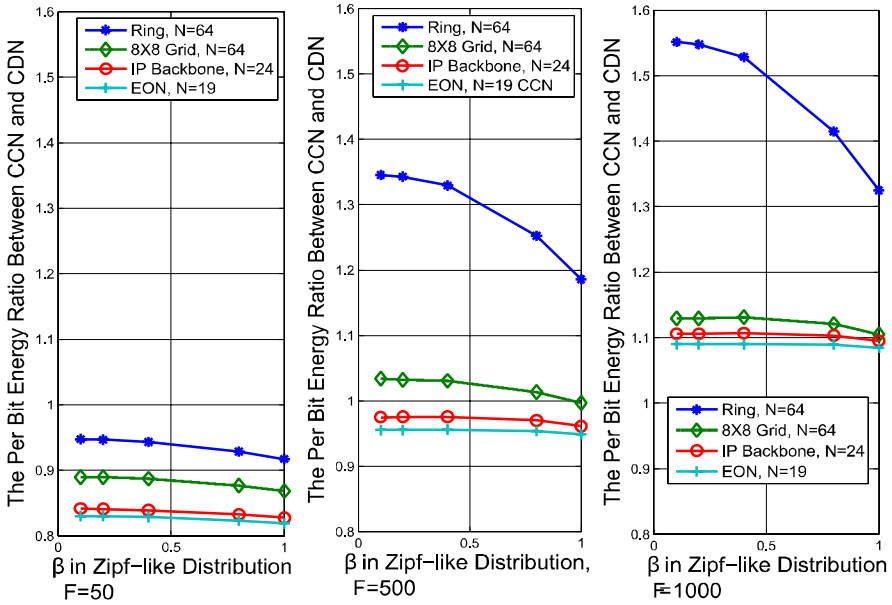


Figure 6: The ratio of optimal energy per bit between CCN and conventional CDN [10]

In another more general direction, content replication and replica placement, as a major component of content management in CDNs, has long been investigated with respect to some generic cost metrics (e.g. transmission costs) or with more concrete objectives such as bandwidth and delay efficiency. While the proposed solutions with these objectives are likely to simultaneously achieve energy efficiency together with the explicit objectives targeted by the solution, the extent of achievable energy saving in these solutions is unknown. Only few studies [12] have directly addressed the impact of content replication strategies on the energy consumption behaviour. Jayasundara et. al. [12] studied a Video on Demand (VoD) distribution network with different replica placement strategies. Using macroscopic energy consumption models for the various elements of the VoD architecture and adopting optical technology as the dominant access network in the VoD architecture, they investigated five different replica placement strategies characterized by the proximity to a cluster of end-users. As shown in Figure 7, five different locations were considered for content replication. Location 1 represents a head-end server, location 2 corresponds to a video hob office, location 3 is a video serving office, and finally locations 4 and 5 correspond to Optical Network Termination (ONT) and Optical Line Unit (OLU), respectively. According to Figure 8, none of the scenarios preserves an absolute superiority with regard to all possible demand rates. According to Figure 8 (a), when the demand rate is low, replication and serving the videos from location 1 is the most energy efficient replication strategy. When the demand rate increases, replication in the locations closer to the edge becomes superior in terms of energy saving. This suggests a hybrid replication policy where different portions of contents are replicated in different locations. Figure 8 (b) shows the maximum energy saving that can be achieved by optimizing the portions of replicated contents per location with respect to different demand rates. Figure 8 (c) illustrates optimal replication in the presence of time varying demands as a function of the time of day (ToD). The demand rate is varied with ToD such that the effective download rate can be roughly modelled by a piece-wise Poisson distribution. The figure indicates that the replication of popular contents closer to the end-users is an efficient policy in peak hours, whereas in off-peak hours it becomes inefficient.

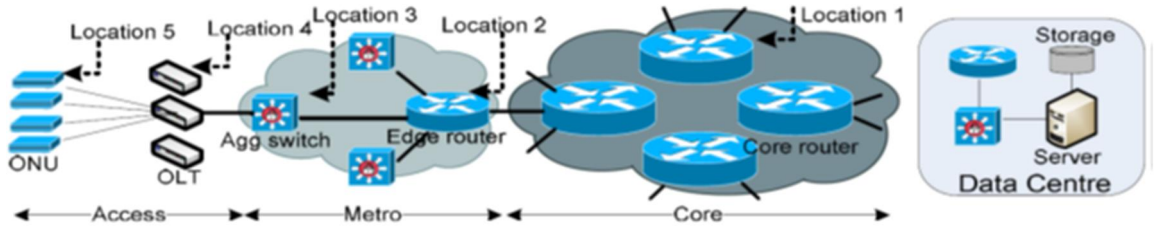


Figure 7: VoD architecture [12]

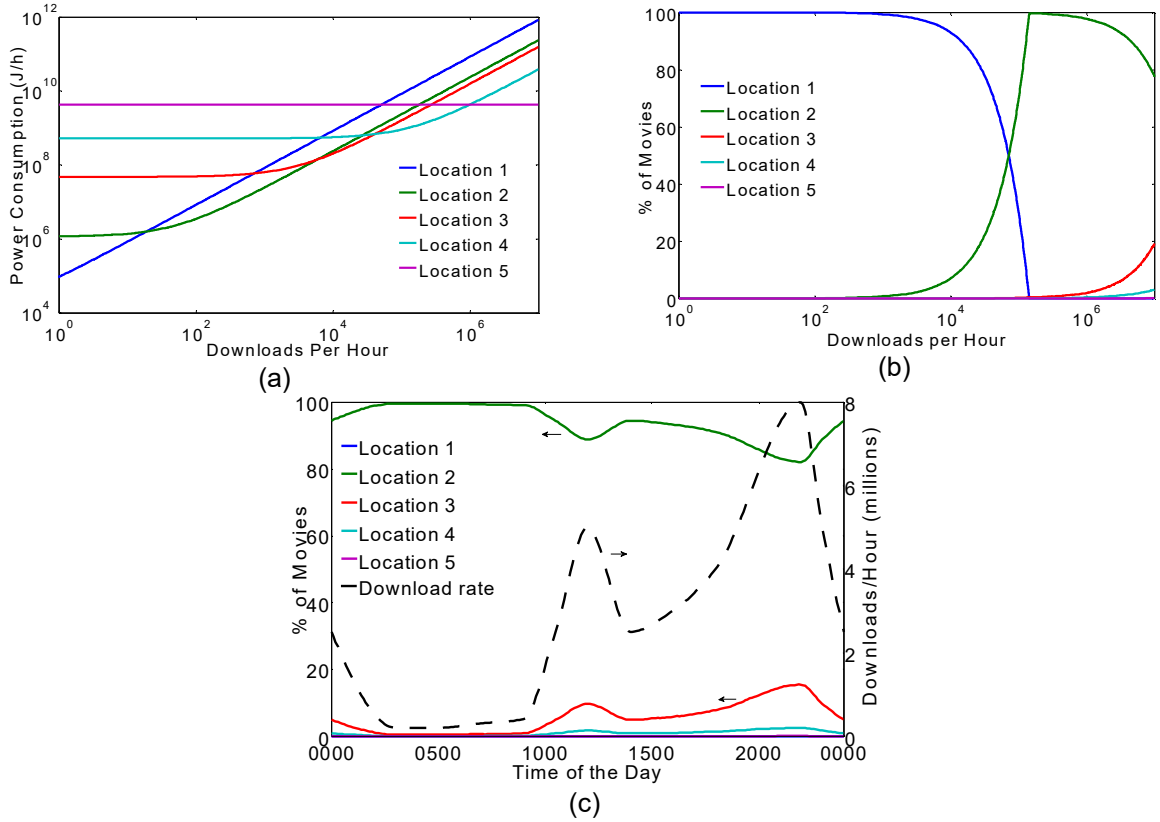


Figure 8: Energy consumption w.r.t: (a) demand rate, (b) demand rate and optimal replication, (c) time variant optimal replication

2.2 CDN Evaluation Measures

In this section we identify some metrics and measures which are used to evaluate the performance of CDNs in the previous studies. For each metric, we provide a summary of the efforts on using it in order to evaluate the performance of CDNs.

2.2.1 Latency and throughput performance

Latency in networking context can be defined by the time it takes for a request to travel from the sender to the receiver and for the receiver to process that request. In other words, the round trip time from the client's browser to the server that a user experience it by sending a request. The most important goal in CDN architecture is also to improve user-experienced performance in terms of latency. Therefore, measuring the delay performance is necessary in assessing a CDN. However, the detailed definition and assessment of latency can be different from network to network by applying different assumptions. In [13] two major CDN's delay components are exposed to discuss. First, DNS resolution delay, that is the time for the CDN's internal DNS system to supply the client the address of the "best" CDN content server; and second, the content-server delay, that is the round-trip time between client and selected CDN server. An enhanced King [14] approach launched from a large number of vantage points is used. The delay performance perceived by Akamai and Limelight customers worldwide have been quantified for both DNS resolution and content serving. These two CDNs are compared with regards to the numbers of their content servers, their internal DNS designs, the geographic locations of their data centers, and their DNS and content server delays. Table 1, Table 2 and Figure 9 show the results of comparison.

Table 1 - Delay Comparison between Akamai and Limelight

	akamai.net		akamaiedge.net	limelight	
delay (ms)	DNS	server	DNS	DNS	server
avg.	40.33	28.75	52.15	43.91	81.86
50%	15.01	8.92	20.58	17.13	43.15
90%	85.40	55.76	107.79	138.50	168.02
95%	138.35	103.38	189.29	170.22	221.74

Table 2 - Details of Delay Performance Breakdown

	Akamai		Limelight	
delay (ms)	DNS	server	DNS	server
North America	115.81	67.24	78.64	79.03
Europe	131.08	82.54	103.34	110.05
Asia	122.5	125.53	199.84	284.4
Oceania	163.68	173.17	279.12	266.66
South America	402.35	312.52	388.17	368.66
Africa	961.03	647.08	596.03	591.45

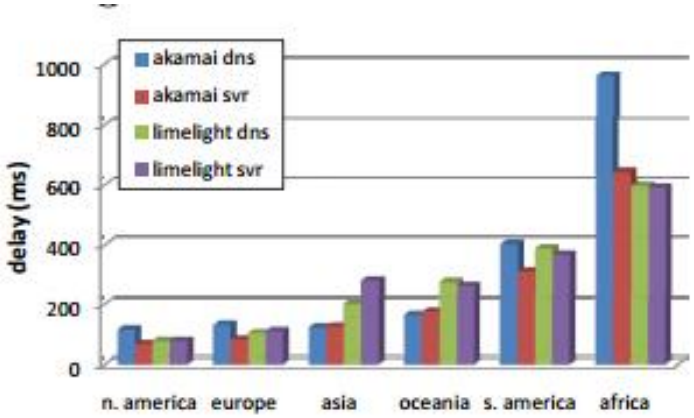
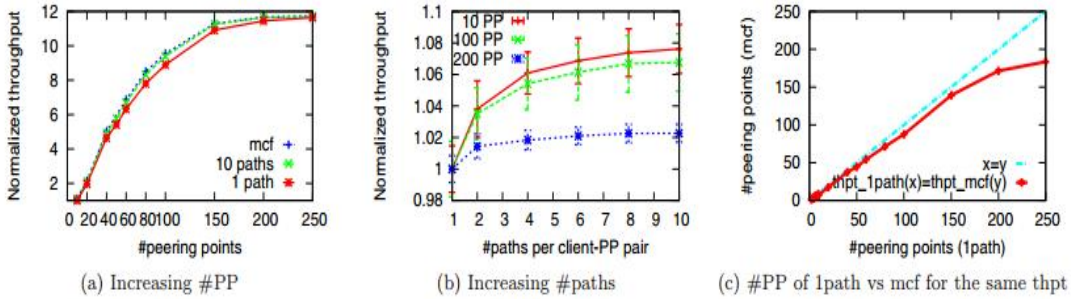


Figure 9 - Delay Performance Breakdown (by continent)

By measuring latencies from Google CDNs, the analysis is performed based on RTT logs, BGP tables, and Netflow records obtained in 2 days [15]. The main result is that redirecting every client to the server with least latency does not suffice to optimize client latencies. In addition, the authors also find that queueing delays often override the benefits of a client interacting with a nearby server. A system is built called WhyHigh, which measures client latencies across all nodes in the CDN and diagnoses the latencies using active measurements such as traceroutes and pings, in combination with datasets such as BGP paths and flow records.
In order to decrease the latency of CDN, [16] proposes an idea which serves the objects in a web page that have the largest impact on page latency out of the closest or fastest caches in the hierarchy. They present schemes for identifying these objects and develop mechanisms to ensure that they are served with higher priority by the CDN, while balancing traditional CDN concerns such as optimizing the delivery of popular objects and minimizing bandwidth costs. To establish a baseline for evaluating improvements in page latencies, they collect and analyze publicly visible HTTP headers that reveal the distribution of objects among the various levels of a major CDN's cache hierarchy. Through extensive experiments on 83 real-world web pages, latency reductions of over 100 ms can be obtained for 30% of the popular pages, with even larger reductions for the less popular pages. Using anonymized server logs provided by the CDN, the feasibility of reducing capacity and staleness misses of critical objects by 60% with minimal increase in overall miss rates, and bandwidth overheads of under 0.02%.

Beside latency, throughput is another metric related to response time, which indicates how much data can be transmitted in a given amount of time, with a high amount being desirable. In [17] two ways are mentioned to improve the CDN throughput: (1) use path selection and multipath routing to avoid network bottleneck between a CDN server and a client, and (2) increase the number of peering points of the CDN [17]. The study follows this question: What are the benefits of optimal path selection– beyond increasing the number of peering points– to improve the aggregate throughput of a CDN? It models two CDN design choices; in CDNs, it increases peering point at the edge, and, in ISPs, it improves path selection at the core, and compares the result of two models (Figure 10). The results show that adding new peering points helps CDNs improve the throughput most. On the other hand, ISP-CDNs could not benefit much from their ability to optimize the routes. The study also compare two ways to improve peering point in order to increase throughput: the more distributed CDNs (e.g., Akamai) increase the number of server locations to increase peering points; the more centralized CDNs (e.g., Limelight) only deploy servers at a few locations but deploy many peering points at each location. Their evaluation shows that the CDN design should be more centralized with many peering points, because it requires lower operating cost than the more distributed approach to achieve the same throughput.



(a) Increasing #PP (b) Increasing #paths (c) #PP of 1path vs mcf for the same thpt

Figure 10 - Multipath has little throughput improvement over increasing #server locations (power law graph)

As the performance of CDNs is difficult to characterize due to dependence on different factors, authors in [18] develop a methodology called DBit that can determine whether one CDN's user-perceived performance is statistically different from another. They focus on latency to compare the throughput of CDNs. DBit has three stages. First, it obtains active measurements of CDNs from a set of vantage points V. In the second stage, it compares if the distribution of measurements in CDN A is better than CDN B. Finally, in stage three, DBit uses the Binomial test to determine if the fraction of nodes with cA,B,v bits being 1 is statistically significant. If, for a significant number (as determined by the Binomial test) of nodes, cA,B,v is 1, then we conclude that A is indeed faster than B. In their study, they collect data from three Photo CDNs: Google+, Facebook and Flickr, using 162 PlanetLab vantage points each from a distinct site. From each PlanetLab vantage point, they measure two forms of latency: cold-fetch latency (the time taken to download a non-cached photo) and hot-fetch latency (the time taken to download a cached photo).

Table 3 - Hot and cold fetch results for Flickr against other Photo CDNs. Ticks denote a significant comparison, the coverage includes the total set of (∧ sign denotes a "faster than" relation)

		Global Hot Fetch Comparison					
		Flickr Vs Facebook		Flickr Vs Google+		Flickr Vs Akamai	
		FB ∧ Fl	Fl ∧ FB	G+ ∧ Fl	Fl ∧ G+	Ak ∧ Fl	Fl ∧ Ak
Global	$c_{A,B,v}$ frac.	0.6481	0.1296	0.9444	0	0.8086	0
	P-value	2.0e-04	5.1e-23	6.1e-35	3.4e-49	7.9e-16	3.4e-49
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓	
		N. America Hot Fetch Comparison					
N. America	$c_{A,B,v}$ frac.	0.7761	0.0149	0.9701	0	0.9104	0
	P-value	6.5e-06	9.2e-19	3.1e-17	1.4e-20	1.5e-12	1.4e-20
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓	
		Europe Hot Fetch Comparison					
Europe	$c_{A,B,v}$ frac.	0.5949	0.2278	0.962	0	0.7595	0
	P-value	1.2e-01	1.3e-06	2.7e-19	3.3e-24	4.2e-06	3.3e-24
	Significant?	✗		Google+ ✓		Akamai ✓	

		Global Cold Fetch Comparison					
		Flick Vs Facebook		Flickr Vs Google+		Flickr Vs Akamai	
		Fb ∧ Fl	Fl ∧ FB	G+ ∧ Fl	Fl ∧ G+	Ak ∧ Fl	Fl ∧ Ak
Global	$c_{A,B,v}$ frac.	0.9074	0.0185	0.9691	0.0062	0.8519	0.0123
	P-value	2.1e-28	2.4e-43	3.1e-40	5.6e-47	1.2e-20	4.5e-45
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓	
		N. America Cold Fetch Comparison					
N. America	$c_{A,B,v}$ frac.	0.9403	0	0.9403	0	0.806	0.0149
	P-value	1.1e-14	1.4e-20	1.4e-20	1.1e-14	4.5e-07	9.2e-19
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓	
		Europe Cold Fetch Comparison					
Europe	$c_{A,B,v}$ frac.	0.8734	0.0253	0.9873	0.0127	0.8734	0.0127
	P-value	5.5e-12	1.1e-20	3.3e-24	2.7e-22	5.5e-12	2.7e-22
	Significant?	Facebook ✓		Google+ ✓		Akamai ✓	

Table 3 shows the results of both the second and third stage of DBit for Flickr in comparison to other Photo CDNs for both hot and cold fetches. The cA,B,v row shows the total fraction of nodes where the hypothesis is true, whereas the p-values show if the fraction of true nodes is significant. Since a majority of the comparisons reveals statistically significant differences in performance between Flickr and other Photo CDNs, the conclusion is that for the given dataset, Flickr's performance is indeed inferior.

2.2.2 Utility

Utility captures the traffic activity in a CDN, expressing the usefulness of surrogate servers in terms of data circulation in the network. In particular, the CDN utility is a metric that shows the relation between the numbers of bytes of the server content against the number of bytes of the pulled content. In another word, utility refers to the quantification of a CDN's traffic activities and represents the usefulness of its replicas in terms of data circulation in its distributed network. The CDN utility is evaluated under different network topologies, traffic models and Web site models [19]. Net utility u_i of a CDN surrogate server *i* is quantified by equation (1) and considering that a CDN has N surrogate servers, the CDN utility *u* can be defined by Equation (2).

$$u_i = \frac{2}{\pi}\arctan(\xi) \quad (1) \qquad u = \frac{\sum_{i=1}^{N} u_i}{N} \quad (2)$$

where the parameter ξ is the ratio of the uploaded bytes to the downloaded bytes. The resulting net utility ranges to [0,1]. The value $u_i = 1$ is achieved if the surrogate server uploads content only (i.e. ξ = infinity). On the contrary, the value 0 is achieved if the surrogate server downloads content only. To study which parameters affect the CDN utility, the authors have performed four set of experiments including: investigating the impact of the network topology backbone on the CDN utility, the impact of various popularity distributions, the impact of correlation between objects popularity and objects size, and finally the impact of various CDN redirection policies.

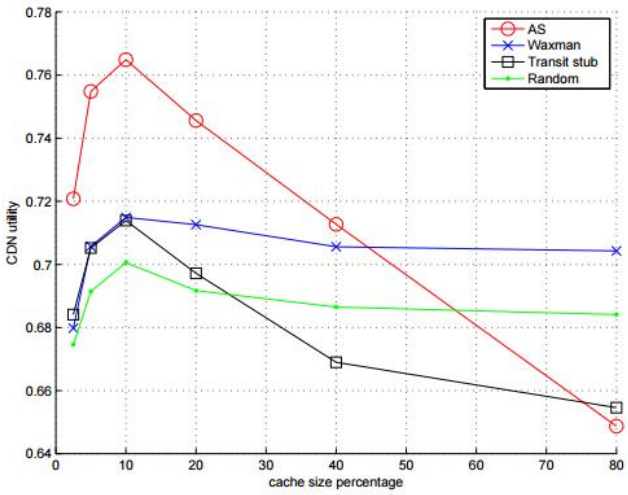


Figure 11- - CDN utility vs. Network topology

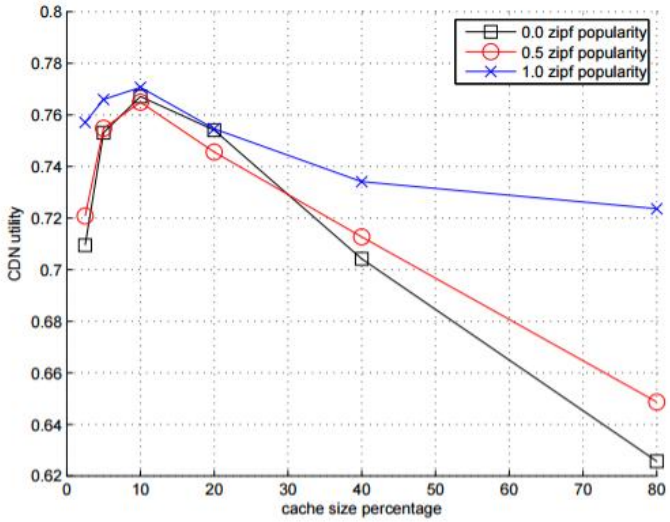


Figure 12 - CDN utility vs. Popularity distribution

The simulation results showed that a performance peak, in terms of CDN utility has been detected. The peak is invariant of the network topology, the traffic model and the Web site model.

CDN utility - a metric that captures the traffic activity in a CDN, is also exploited in [20]. It addresses the content replication problem by replicating content across a geographically distributed set of servers and redirect users to the closest server in terms of CDN utility. The utility of content delivery is measured in [21] via MetaCDN (a system that exploits storage cloud resources, creating an integrated overlay network that provides a low cost, high performance CDN for content creators [22]) with capturing the system-specific perceived benefits. This utility measure is used to devise a request redirection policy that ensures high performance content delivery. Also, it quantifies a content provider's benefits from using MetaCDN based on its user perceived performance.

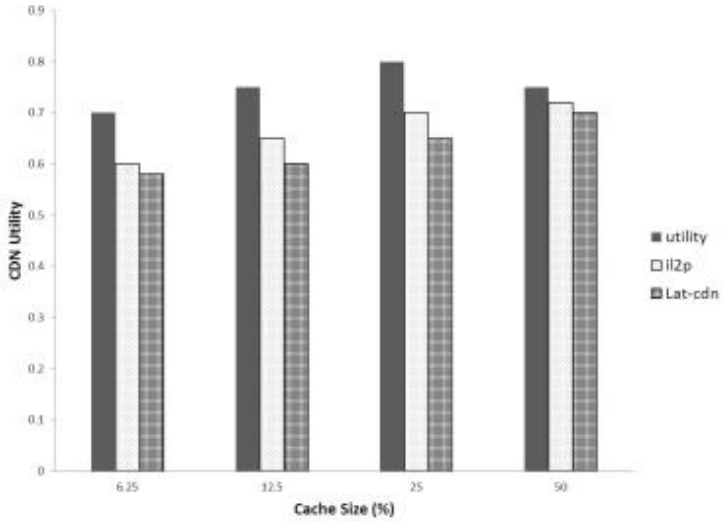


Figure 13 - CDN Utility vs Cache Size

This algorithm achieves low replica redundancy. This is very important for CDN providers since low replica redundancy reduces the computing and network resources required for the content to remain updated. Also, it achieves a performance peak in terms of CDN utility at a certain small cache size.

Another utility model is introduced in [23] which measures the content-serving ability of the peering CDNs system for different traffic types using the equations 1 and 2. Peering CDNs as the interconnection of distinct CDNs is one of the possible solutions to handle flash crowds, Web resources over-provisioning, and adverse business impact, by providing coordinated and cooperative content delivery among CDNs. Peering between CDNs can be established for a short or long period to

handle workload variations, thus allowing providers to expand their reach and capacity. In this model the authors show that although the peering CDNs system observes high utility in terms of satisfying content requests, its content serving ability is largely dependent on the participating providers' utilities. A schematic representation of the methodology used to evaluate the utility of peering CDNs is provided in Figure 14.

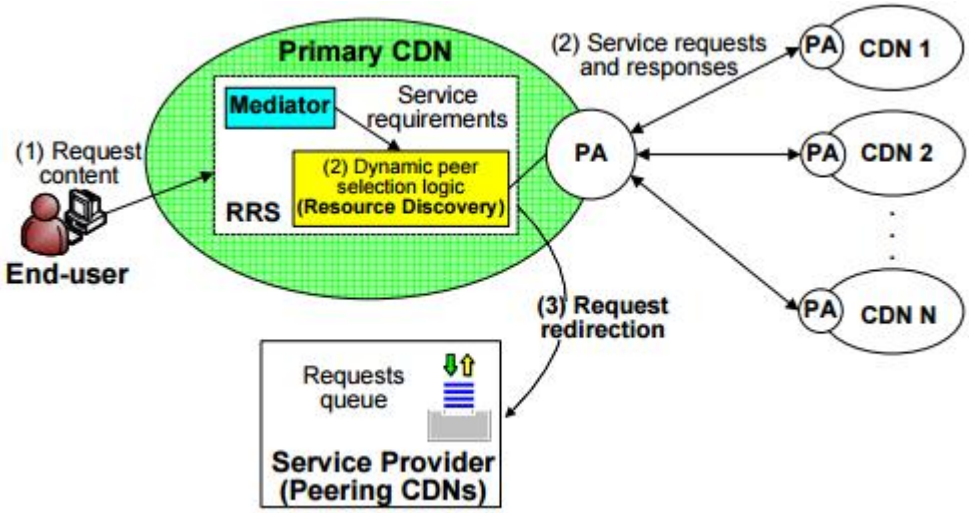


Figure 14 - Simulation methodology

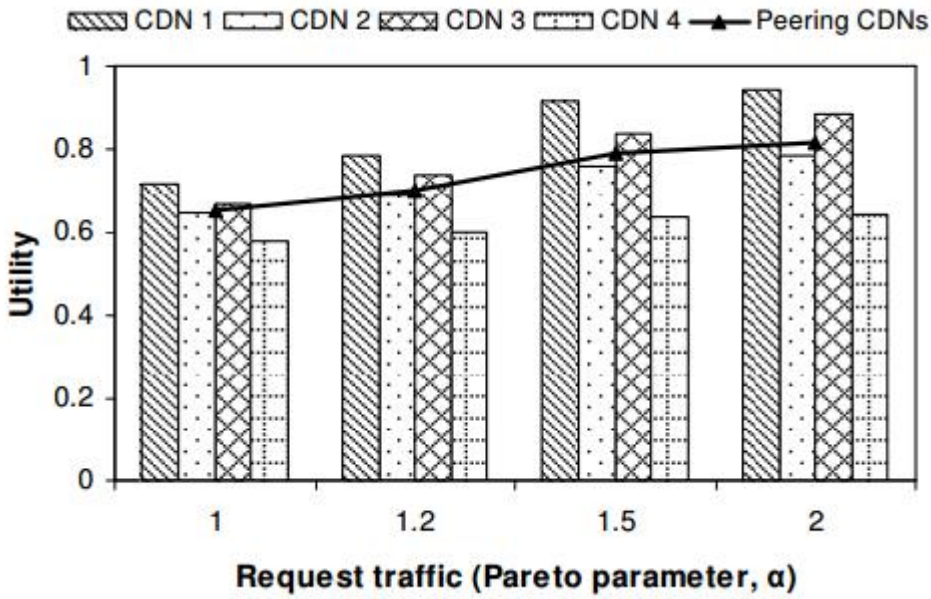


Figure 15 - Utility measures for different traffic types

Figure 15 shows that CDN 1 (primary) and CDN 3 (a peer) demonstrate higher utility than that of other peers. In this case, they contribute more servers (with higher capacity) to the system than other peers. In addition, server selection results in more requests to be redirected from CDN 1 to the servers of CDN 3, identifying it as a peer with close proximity to the primary. Figure 16 shows the total completions (i.e. the number of completed requests) in each participating CDN and in the peering CDNs system.

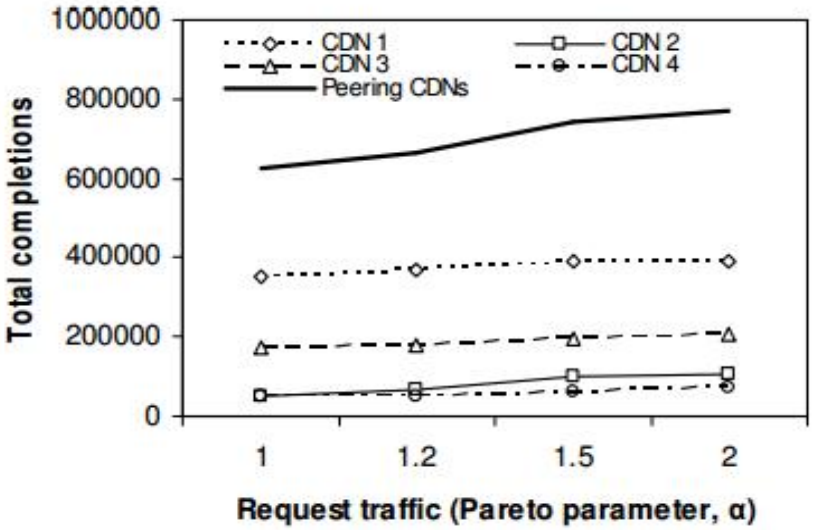


Figure 16 - Total completions in each CDN and in the peering CDNs system

The result of the impact of completions on the utility of peering CDNs is presented in Figure 17. The total number of completions and the resultant utility decreases for highly variable traffic types. The observed trend is sensible as the number of served requests (completions) and associated rejections act as the major parameters in the proposed utility model.

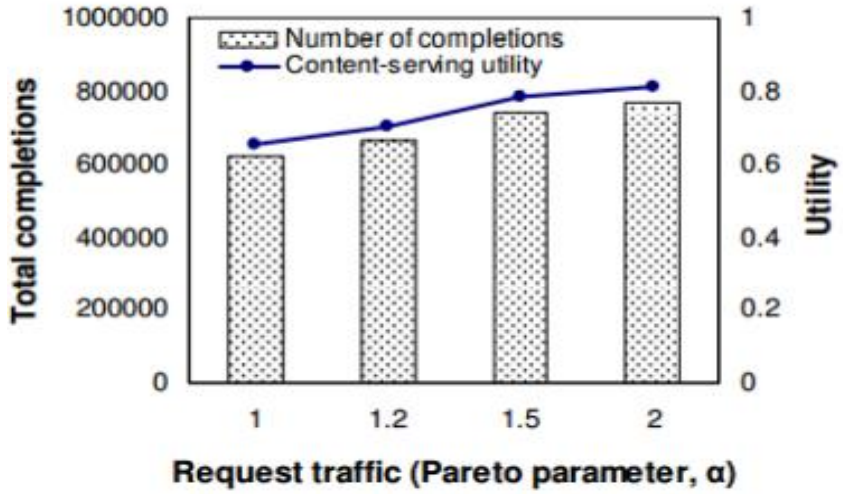


Figure 17 - Total completions and utility for peering CDNs

In order to optimize video service delivery, a utility-based approach is introduced in [24], which uses a global utility function for each actor in the video service delivery chain, including content provider, operator, and client.

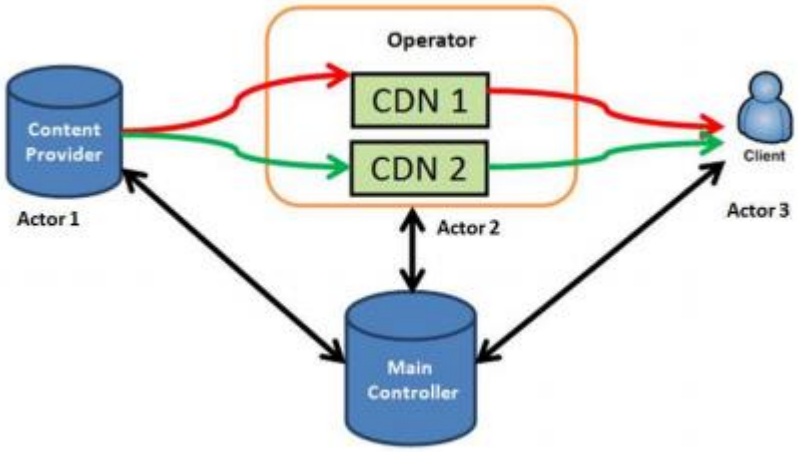


Figure 18 - Main three actors in the video chain

The utility function $1 - e^{x}$ is used, where x is the parameter which can be: i) positive value like throughput, available hardware (high values are better), and ii) negative values like cost and networks load (low values are better). They consider the cost of transmitting the content in the network as a parameter, and compute the utility score of each actor. The validation is done in two ways. First, using simulation to optimize the parameters of utility function and second, validation by using software defined network controller through the SDN implementations based testbed.

2.2.3 Function of load

Load balancing aims to optimize resource usage, maximize throughput, minimize response time, and avoid overload of a CDN server. In [25] some techniques to turn off CDN servers during periods of low load are proposed, where the aim is to seek a balance among three key design goals: maximize energy reduction, minimize the impact on client-perceived service availability (SLAs), and limit the frequency of on-off server transitions to reduce wear-and-tear and its impact on hardware reliability. The standard linear model is used to measure the power consumed by a server serving a particular load and then based on that they performed local/global load balancing in the CDN. The proposed mechanisms are evaluated using real production workload traces collected from a large commercial CDN over 25 days from 22 geographically distributed clusters across the US. They evaluate different algorithms and the impact of different parameters. The evaluation using real production workload traces from a large commercial CDN shows that it is possible to reduce the energy consumption of a CDN by more than 55% while ensuring a high level of availability that meets customer SLA requirements with only a modest number of on-off transitions per server per day (Figure 19 and Figure 20).

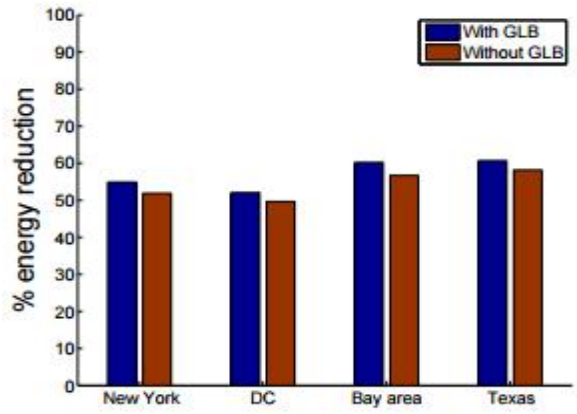


Figure 19 - % Energy reduction

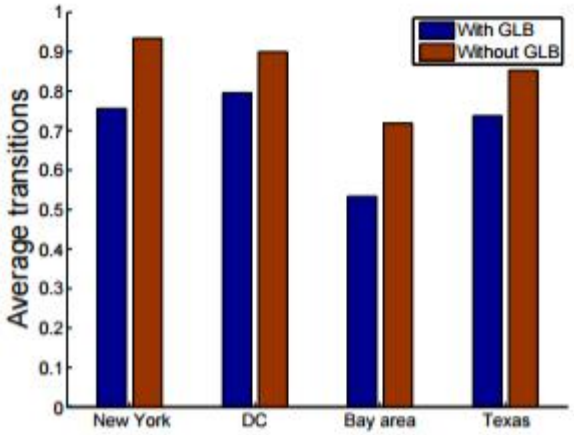


Figure 20 - Average Transitions (per server per day)

To overcome the major limitations of Commercial and Academic models, a new load balancing (LB) algorithm for the efficiency analysis of CDN surrogates is proposed in [26]. A new architecture that make use of the existing resources of common Internet users in terms of storage space, bandwidth and Internet connectivity to create a Distributed Content Delivery Network (DCDN). The LB mechanism ensures that common internet users with limited computing resources are not burdened with heavy load and requests are fairly assigned to them. Three different scenarios of DCDN (Table 4) are created and compared with commercial CDN.

Table 4 - Simulation Setup

	Commercial CDN	DCDN: Scenario 1	DCDN: Scenario 2	DCDN: Scenario 3
Number of Clients	150	150	75	30
Number of Surrogates (or Servers)	3	6	6	6
Link Capacity (Mbps)	100	10	10	10
Load Balancing Algorithm	round robin	server load	server load	server load

Performance evaluation of DCDN architecture using simulation techniques shows that proposed architecture (with LB) can indeed offer same or even better performance compared to commercial CDN in certain cases (Figure 21 , Figure 22).

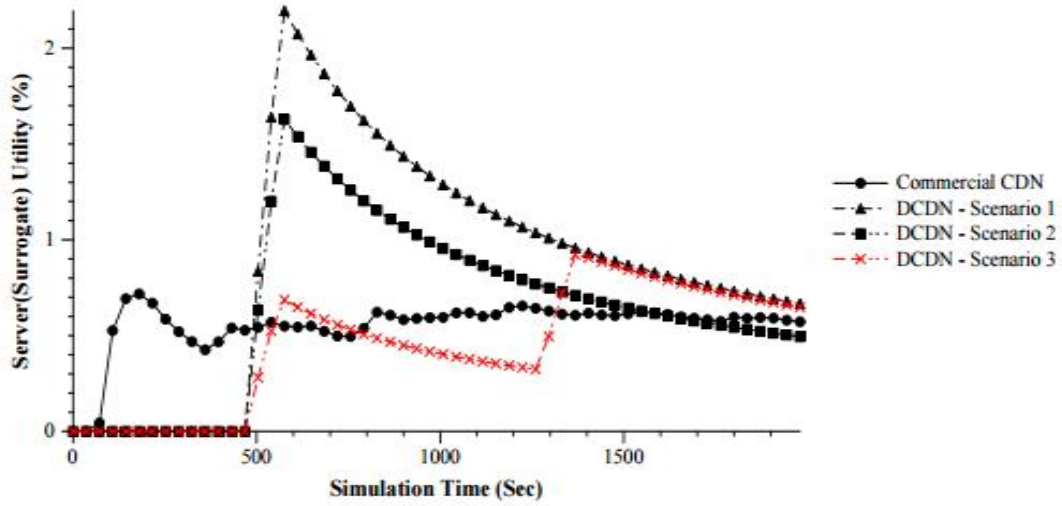


Figure 21 - DCDN Surrogate (Server) Utilization

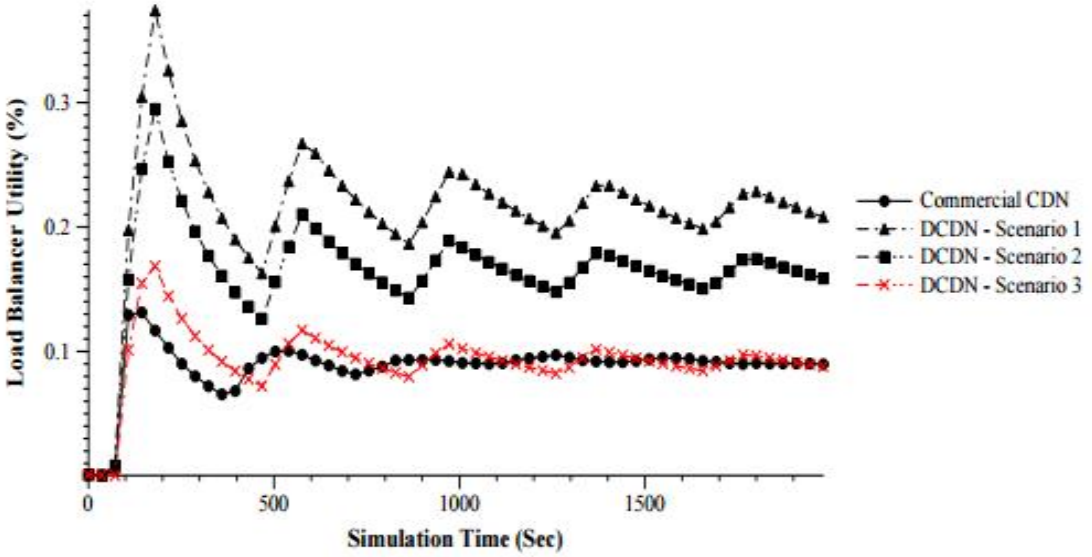


Figure 22 - DCDN Server (Load Balancer) Utilization

2.2.4 Server location

Server location is a very important feature since locating a server near to the requesting client not only eliminates the distance that content travels, but also reduces the number of hops a data packet must make. The result is less packet loss, optimized bandwidth and faster performance, which minimizes time-outs, latency and jitter, while improving overall user experience. In an especial case study, the co-location approach to CDN platforms adopted by Akamai is investigated [27]. It tries to deploy servers in numerous Internet locations, brings inherent performance benefits over a more consolidated data centre approach employed by other influential CDNs such as Limelight.

2.2.5 Number of hops

Number of hops is an effective factor on the user perceived latency. Minimizing the number of hops can be described as a problem of optimally replicating objects in CDN servers. In [28] each autonomous system (AS) is considered as a node in a graph with one CDN server with finite storage capacity for replicating objects. The optimization problem is to replicate objects so that the average number of ASs traversed is minimized when clients fetch objects from the nearest CDN server containing the requested object. In other word, the goal is to minimize the average number of inter-AS hops that a request must traverse.

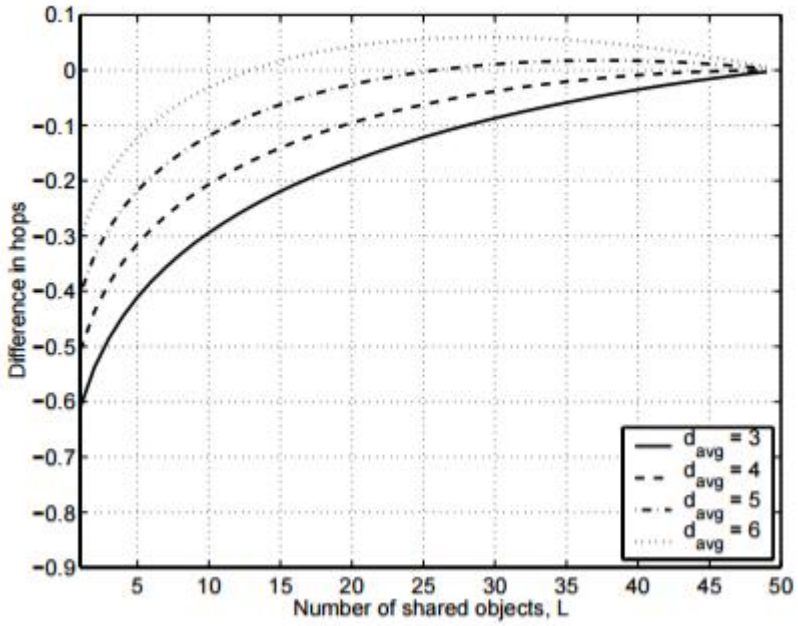


Figure 23 - Gain from cooperation for K =50, 1000 objects

In the aforementioned study, four natural heuristics are developed (Figure 23) and compared numerically using real Internet topology data. The results show that the best performing heuristic is Greedy-Global which has all the CDN servers cooperating. The difference in performance between Greedy-Global and the simpler heuristics was up to 24%.

Authors in [29] propose a lifetime-based greedy caching approach (LBGC) which involves both caching decision and cache replacement. Contents are preferred to be stored in nodes with high betweenness centrality, and only expired cache items can be replaced by new arrival contents, which differs from least recently used (LRU) strategy and guarantees the caching of popular contents. The effectiveness of different caching strategies are compared across a range of cache sizes, from 0 to 1000 chunks (0% to 10%) in Figure 24.

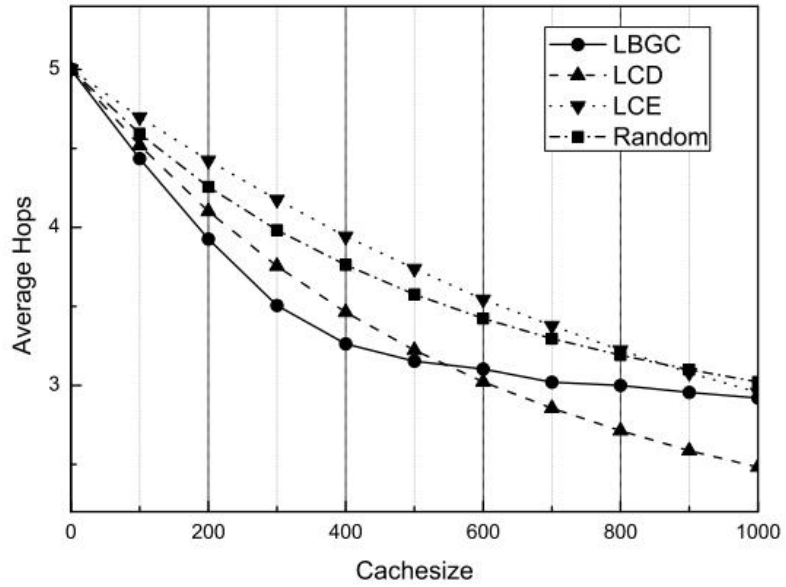


Figure 24 - Cache size vs. average hops

All the schemes provide performance improvement as the cache size increases. When the cache size is lower than 500 chunks, the proposed strategy outperform others, while for larger cache sizes it

leads to performance degradation. Simulation results indicate that LBGC can improve the cache hit ratio and reduce average hops to get content items compared with other caching approaches.

2.2.6 Hit Rate

The surrogates are normally proxy caches that serve cached content directly with a certain hit ratio; the un-cached content is initially obtained from the origin server before responding. A general expression for a content distribution environment is proposed in [30] and the performance impact of design variables such as caching hit ratios, network latency, number of surrogates, and server capacity is studied.

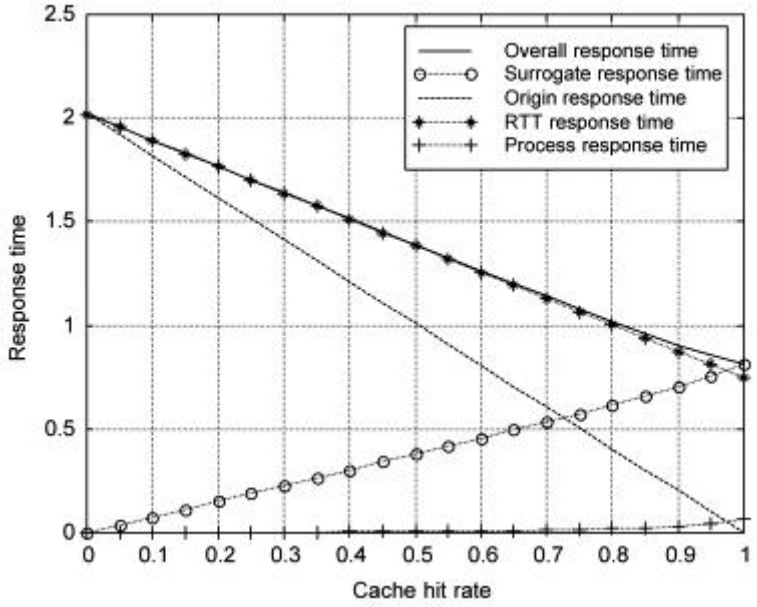


Figure 25 - Cache hit rate vs. Response time

The result (Figure 25) depicts that, under reasonable conditions, the roundtrip time follows a linear function dependent on the caching hit ratio with a negative slope. In addition, if the servers are dimensioned with a capacity value dependent on just the traffic arrival rate, the caching hit ratio does not have any effect on the process time. A trade-off between roundtrip time and process time determines the scenario for optimum values of total response time within a caching hit range.

2.3 Energy Consumption in CDNs

2.3.1 Analysing CDN energy consumption as a function of the number of Surrogate Servers

In [31] Authors represent a CDN as a main server, named primary server, storing the whole data set, connected to several surrogate servers which are positioned on network edge, closer to end users. Surrogate servers store content depending on their cache size and on content popularity, possibly estimated among end users close to the surrogate servers. As the real Internet map is difficult to be estimated, they defined a three-tier graph and model a typical CDN network with one primary server that maintains all the contents in the data set, and a variable number of surrogate servers (Figure 26). To compute total energy consumption of the CDN, four different energy consumption components, namely storage, server, transmission and synchronization are defined.

$$E_{tot} = E_{storage} + E_{server} + E_{synch} + E_{tx} \qquad (1)$$

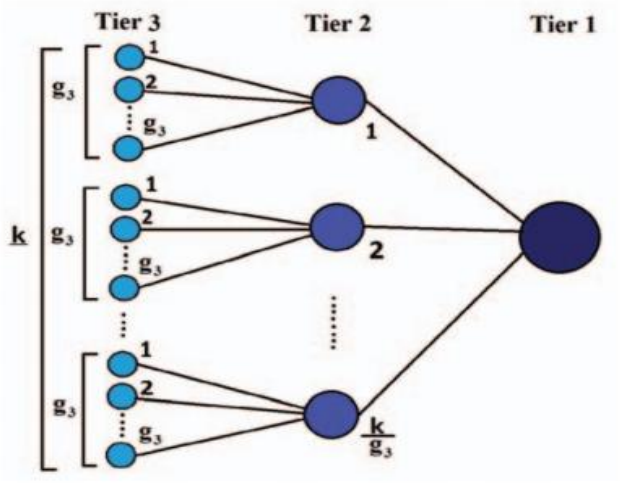


Figure 26 - Simplified map

- ***Storage energy consumption*** is the energy consumed to store the whole data set in all the servers.

$$E_{storage} = \sum_{m} Bn_m P_{st} t \qquad (2)$$

- ***Server energy consumption*** is the energy consumed by the server for each received request to process it, get the content and send it.

$$E_{server} = \sum_{m} Br_m E_{sr} \qquad (3)$$

- ***Synchronization energy consumptio***n is the energy consumed to propagate modified content to the proper surrogate servers.

$$E_{synch} = \sum_{m} Bm_m n_m [E_r(H_{ps} + 1) + E_l H_{ps}] \qquad (4)$$

- ***Transmission energy consumption*** is the energy consumed to transmit content to the user that has requested it.

$$E_{tx} = P_A \sum_{m} Br_m [E_r(H_{sd}^A + 1) + E_l H_{sd}^A] + P_B \sum_{m} Br_m [E_r(H_{sd}^B + 1) + E_l H_{sd}^B] + P_C \sum_{m} Br_m [E_r(H_{sd}^C + 1) + E_l H_{sd}^C] \qquad (5)$$

Table 5 – Notations and Parameter Setting

Symbol	Default value	Description
M	1000	total number of contents in the data set
B	10^6 bits	size of each content
t	6000s	time period of the analysis
n_m	Sc.S	Number of surrogate servers . cache size
r_m	100,1000	requests for content m
m_m	10,100	modifications to content m
H_{sd}^A	3	hops to fetch content from the same Tier 3 ISP
H_{sd}^B	14	hops to fetch content from the same Tier 2 ISP
H_{sd}^C	25	hops to fetch content from the core network
P_{st}	$7.84 \cdot 10^{-12}$ W	storage power consumption per bit
E_r	$1.2 \cdot 10^{-8}$ J/bit	router energy consumption per bit
E_l	$1.48 \cdot 10^{-9}$ J/bit	link energy consumption per bit
E_{sr}	$2.81 \cdot 10^{-7}$ J/bit	server energy consumption per bit

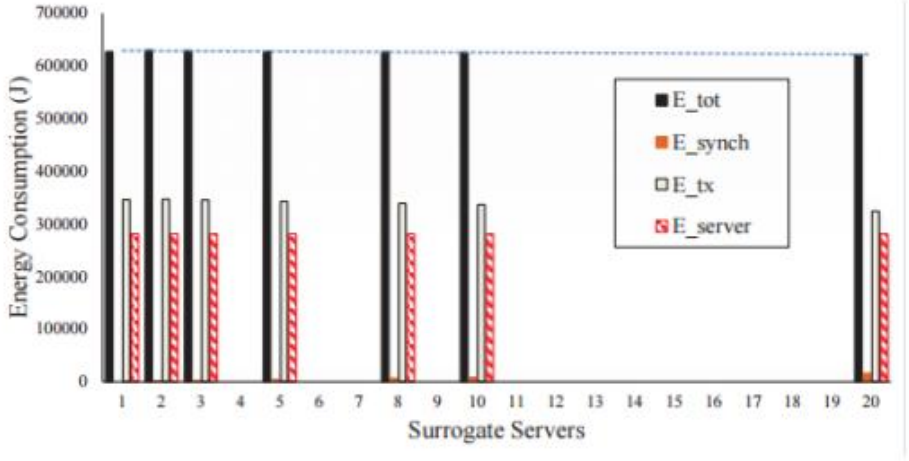


Figure 27 - Energy consumption components. Uniform caching. SC = 40%, mm/rm = 0.01, mm = 10, rm = 1000

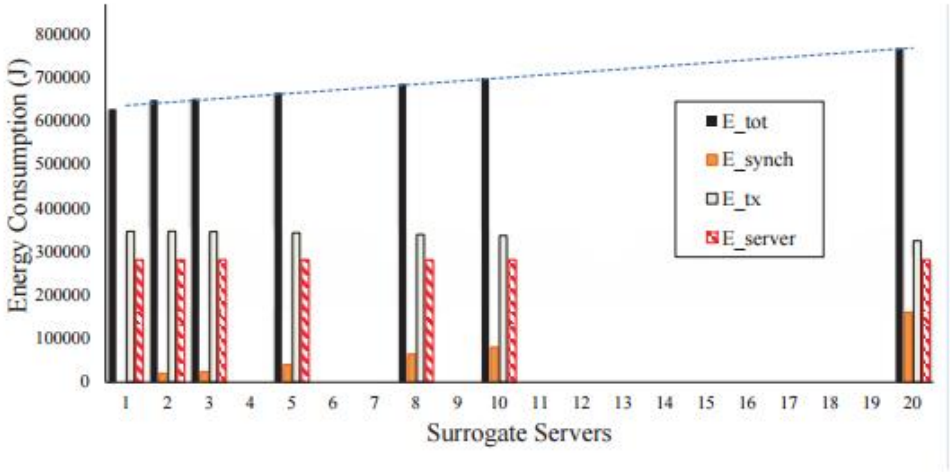


Figure 28 - Energy consumption components. Uniform caching. S_C = 40%, m_m/r_m = 0.1, m_m = 100, r_m = 1000

The results, depicted in Figure 27 and Figure 28, represent the breakdown of energy consumption for each component: total, synchronization, transmission and server energy consumption. The storage energy consumption is not reported, due to its negligibility compared to the other components. The total amount of data in the system is assumed to be equal to M = 1000 content pieces, each with a size of B = 106 bits. The power consumption of servers to store a single bit of data is equal to P_{st} = $7.84 \cdot 10^{-12}$ W [32], which is low in comparison with other energy components.

In conclusion, the total energy consumption does not always decrease by adding more surrogate servers in a CDN network, depending on the ratio between the number of content modifications and the number of content requests. If this ratio exceeds a given threshold (around 0.013 for uniform and 0.028 for popularity-based caching policy), increasing the number of surrogate servers may increase the total energy consumption of the network, due to the increase in the synchronization energy consumption.

2.3.2 Analysing CDN energy consumption as a function of router caches

The impact of using in-network cashes and content delivery network (CDN) cooperation on an energy-efficient routing is also investigated [33]. The aim is to find a feasible routing, so that the total energy consumption of the network is minimized subject to satisfying all the demands and link capacity. Cache hit rate and cache power usage are considered as parameters. To model the problem, aggregated traffic between cities is expressed as a matrix. The matrix represents not only cities, but also content providers. The goal is to find a feasible routing in this matrix such that it satisfies all demands and minimizes the total energy consumption of the network. The objective function is:

$$\min(total\ energy\ consumption) = \min \sum_{\{u,v\}\in E} x_{uv} + \sum_{v\in V} \beta\gamma y_v + \sum_{v\in V} \beta(1-\gamma) z_v \quad (1)$$

As the goal is to turn off links and caches in order to minimize the amount of energy, in above formula x_{uv} indicates the link $\boldsymbol{uv}$, β is the most units of energy used by cache, γ is a fraction of β used by an idle cache, y_v if the cache at router v is turned on or off, and z_v is the load of the cache in router v. The empirical results are presented by studying the impact of cache, CDN and traffic parameters. l_c in the figures indicate the unit of energy by considering that each link uses one unit of energy.

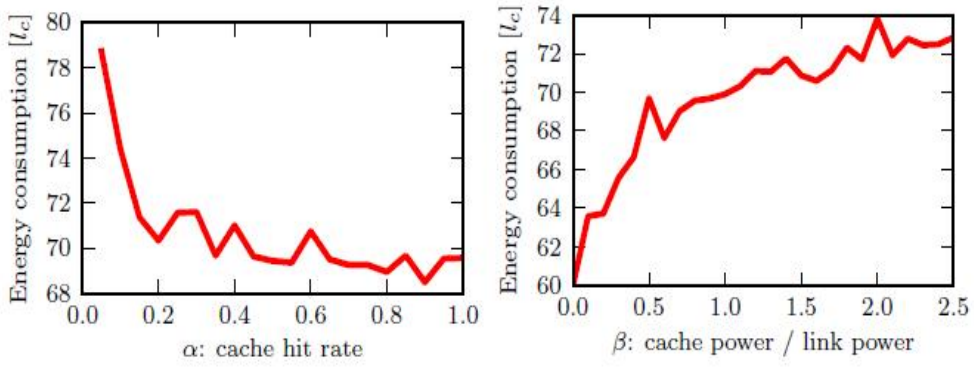


Figure 29- Energy consumption in network varied by cache parameters

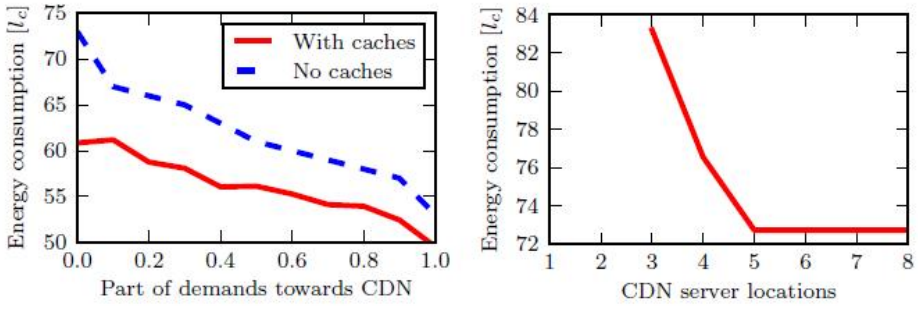


Figure 30- Energy consumption in network varied by CDN parameters

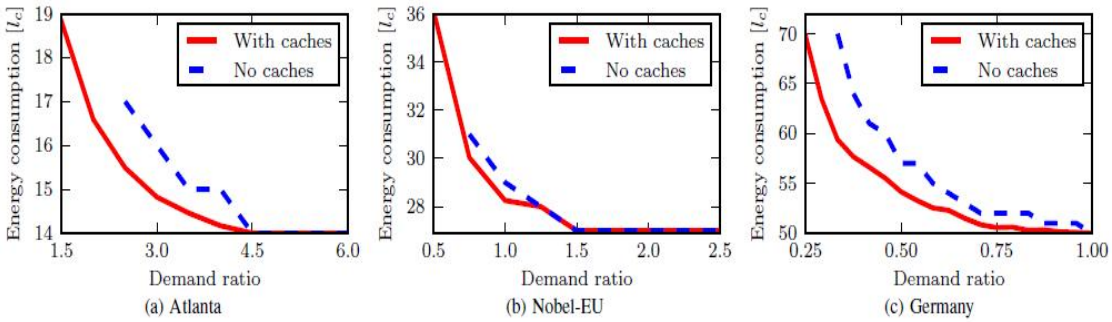


Figure 31 - Comparison of energy consumption with and without caches in the model

Consequently, the proposed model saves energy in backbone networks by disabling equipment, taking into account in-router caches. The total energy saving found oscillate about 25% for realistic parameters. Part of energy saved solely due to introduction of caches is up to 16% in this instance.

2.3.3 Analysing CDN energy consumption with the focus on content centric networking

Using the benefit of content centric networking (CCN) and dynamic optical bypass is another way to examine the energy consumption of content delivery architectures [10]. The same equipment and devices achieve good scalability in energy consumption by optimizing the placement of content according to its popularity. The objective of this study is to address how to achieve energy efficiency by trading off the consumption of various networking resources. In particular, they evaluate the energy cost of a CCN and optical bypass approach used in CDN architectures. Three energy consumption

models for a CCD, a conventional CDN and a centralized CDN using dynamic optical bypass are set up. The total consumed energy E_{tot}^{CCN}consists of two parts: the transport energy E_{tr} and the caching energy E_{ca}. The transport energy, in turn, includes the energy consumption in the core, edge, and access networks, denoted as E_{tr}^{c}, E_{tr}^{e}, and E_{tr}^{a}, respectively. Based on assumptions, E_{tr}^{a} only adds an identical constant to each case and is not included in the analysis.

$$E_{tot}^{CCN} = E_{tr}^{c}(H_r) + E_{tr}^{e} + E_{ca}(H_r) \quad (2)$$

In a conventional CDN architecture, the total energy consumption E_{tot}^{CDN} consists of that consumed by data storage E_{st}, transport E_{tr}, and servers E_{sr}.

$$E_{tot}^{CDN} = E_{tr}^{c} + E_{tr}^{e} + E_{st} + E_{sr} \quad (3)$$

In dynamic optical bypass, the content is delivered by a server that is attached to a core router and transported across the core network mostly via transparent optical connections. We assume that on average a node is H_r^s hops away from the server. The total energy E_{tot}^{bp}, which consists of the transport, server, and storage energy, can be expressed as:

$$E_{tot}^{bp} = E_{tr}^{c}(H_r^s) + E_{tr}^{e} + E_{st} + E_{sr} \quad (4)$$

Table 6- Notation and values of the key parameters

Symbols	Notations	Values
N	No. of nodes	Multiple values
t	Time duration	3600 seconds
R_k	No. of requests of the kth-most popular content	Multiple values
B_k	Size of the kth-most popular content	Multiple values
p_d^r	Power density of a core router	1.2×10^{-8} J/bit
p_d^{wdm}	Power density of a WDM link	1.48×10^{-9} J/bit
p_d^{oxc}	Power density of an OXC	1.95×10^{-8} J/bit
p_{ca}	Power of caching in router buffer	10^{-9} W/bit
p_d^e	Power density of an Ethernet switch	8.21×10^{-9} J/bit
p_d^g	Power density of a gateway router	1.38×10^{-7} J/bit
p_d^{pe}	Power density of a provider edge router	2.63×10^{-8} J/bit
p_{st}	Power of storage (hard disk)	7.84×10^{-12} W/bit
p_d^s	Power density of a server	2.81×10^{-7} J/bit
n	No. of caching replicas	Multiple values
H_r	Average number of hops to a replica	Multiple values

More details about the formulas are available in [10]. The results show that CCNs are more energy efficient in delivering popular content, while the approach with optical bypass is more energy efficient in delivering infrequently accessed content. The relative energy benefit of CCNs and conventional CDNs is more complicated, since the energy performance also depends on factors such as content popularity and catalogue size. The results suggest that a hybrid architecture – a synergy of CCN, server-based CDN, and dynamic optical bypass architectures – may further improve the energy efficiency especially in serving content with heterogeneous popularity.

2.3.4 Analysing CDN energy consumption with the focus on CDN server utilization

In [34] authors present a simple CDN server utilization model to compute the energy consumption in CDNs. They assume that In CDNs the majority of energy consumption originates in surrogate servers. Each surrogate server consumes a constant quantity of energy just by being turned on. The rest can be considered proportional to the utilization. In this context, they assume energy consumption to be proportional to the ratio of active connections against the maximum simultaneous connections each surrogate server is able to handle. Surrogate server's utilization is used as a parameter in order to measure its energy consumption. First, we calculate the power consumed by the surrogate servers while serving the contents to the clients or to the neighbouring surrogate servers. The power consumed by the surrogate server at a given time can be calculated as follows:

$$P_{s_{[t_i,t_j]}} = P_{idle_s} + \frac{Conn_{s_{[t_i,t_j]}}}{conn_{max_s}}\left(P_{max_s} - P_{idle_s}\right) \quad (1)$$

Where $Conn_{s_{[t_i,t_j]}}$ is the actual number of connections the surrogate server *s* handles between time t_i and t_j and P_{idle_s} is the minimum possible power the surrogate s can consume. In this case when a surrogate server is turned on it is supposed to consume a constant amount of power if it is idle and doesn't have any request to serve i.e. it is completely unloaded.
Based on the assumptions, between time interval t_i and t_j the energy consumption of a surrogate server is calculated as:

$$E_{s_{[t_i,t_j]}} = (t_j - t_i) * P_{idle_s} + (P_{\max_s} - P_{idle_s}) * U_{s_{[t_i,t_j]}} \qquad (2)$$

Where $U_{s_{[t_i,t_j]}}$ is the utilization of a server s between t_i and t_j.

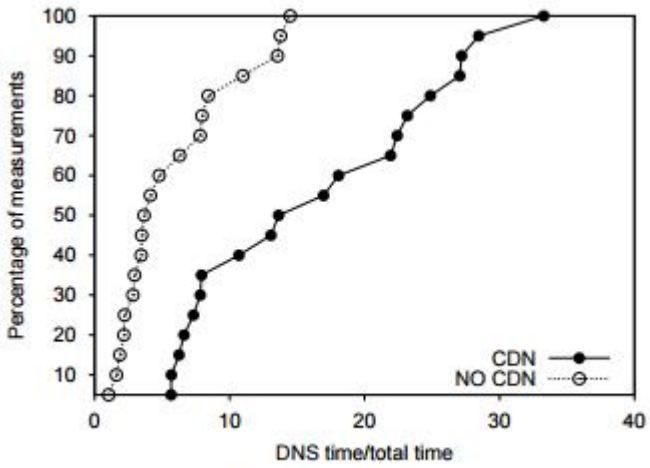
2.4 CDN performance improvements

CDN performance is highly dependent on a number of variables including user location, network, file type and size, time of day and caching. A CDN has multiple replicas of each content item being hosted. It must incorporate dynamic information about network conditions and load on the replicas, routing requests so as to balance the load. Two different CDNs are considered in [35]; Akamai and Digital Island where both CDNs redirect requests using DNS. The same file is fetched from different servers via HTTP and for each fetch, the target IP address, size of returned object (in bytes), and fetch latency were recorded. Their main conclusion is that CDNs provide a valuable service, but neither Akamai nor Digital Island can consistently pick the best server of those available ones, and also these CDNs do not gain very much by choosing an optimal server as by avoiding notably bad servers.
A performance study was conducted over a period of months on a set of CDN companies employing the techniques of DNS redirection and URL rewriting to balance load among their servers [36].

Table 7 - Parallel-1.0 Performance (SEC.) for Server at New and Fixed IP Addresses (JAN.2001)

CDN (DNS TTL in sec.)	*New* Download Time			*New* Completion Time			Fixed IP Download Time		
	Mean	Med.	90%	Mean	Med.	90%	Mean	Med.	90%
Adero (10)	1.15	1.02	1.73	5.40	1.39	9.60	1.09	0.51	1.60
Akamai (20)	1.06	0.34	3.01	1.15	0.39	3.05	1.00	0.41	3.00
Clearway (N/A)	1.19	0.84	2.94	1.19	0.84	2.94	1.16	0.76	3.07
Digisle (20)	1.19	0.47	1.83	1.31	0.52	2.30	1.21	0.43	1.70
Fasttide (230)	1.58	0.96	3.37	2.10	1.19	4.72	1.46	0.91	3.25
Speedera (120)	0.57	0.20	1.18	0.72	0.26	1.53	0.53	0.18	1.01

Table 7 shows the mean, median and 90th percentile values for the new download, new completion (including DNS lookup), and fixed download times for the parallel- 1.0 requests in January 2001. More generally, the results indicate that the use of a DNS lookup in the critical path of a resource retrieval does not generally result in better server choices being made relative to client response time in either average or worst case situations.
To evaluate to what extent the use of a CDN can improve the user-perceived response time, a large set of scenarios with different network conditions and client connections are investigated in [37]. They found that CDNs can offer significant performance gain in normal network conditions, but the advantage of using CDNs can be reduced by heavy network traffic (Figure 32).



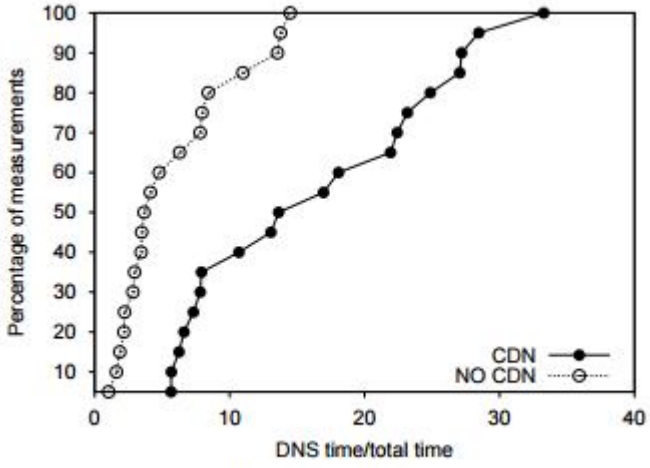


Figure 32 - DNS lookup time vs. page response time

Studying Akamai's (audio and video) streaming service reveals that the current system design is highly vulnerable to intentional service degradations [38]. The results show that it is possible to impact arbitrary customers' streams in arbitrary network regions, not only by targeting appropriate points at the streaming network's edge, but also by effectively provoking resource bottlenecks at a much higher level in Akamai's multicast hierarchy.
Server selection is another way to improve the performance of CDNs which is evaluated in [39], using distance information only available for the ISP. Therefore, with this help, CDN providers are more likely to select servers that offer higher content delivery performance for each particular client.

2.5 Online Social Networks and CDNs

Consumer Internet traffic, largely composed of User generated Content (UGC), uploaded and accessed via online social networking websites, is growing at an impressive rate of 36% per year [40]. CDNs are seen as primary vehicles to help service providers by prefetching and caching content likely to be demanded by their customers. Prefetching and caching content in CDNs can rely on geographical, temporal, or popularity demand patterns. Meanwhile, social networks information can also be a useful source to exploit in the content caching mechanisms of CDNs.
The geographical information is one of the helpful social network's profile information, which helps improve caching mechanisms in CDNs. Exploiting geographical information extracted from social cascades to improve caching of multimedia files in a CDN is discussed in [41]. The proposed approach is validated by using a novel dataset which combines social interaction data with geographic information. The social cascades of YouTube links are tracked over Twitter and a proof-of-concept geographic model of a realistic distributed CDN is built.

In order to find helpful heuristics for content placement, several observations are made based on different datasets from Twitter and other sources in [40]. Also the methodologies for data collection, which can be useful for other researchers working in this space are discussed in detail. These observations are useful to design heuristics to improve CDN performance.
A P2P-assisted video streaming system in social networks, called SocialStreaming is proposed in [42]. They put forward a network coding-based storage strategy and propose a social network-based streaming pre-delivery algorithm in a distributed way, derived from the Metropolis-Hastings algorithm (Figure 33, Figure 34). Also, [43] found that, contrary to popular belief, the simple cache replacement policy LRU outperforms methods based on input from online social networking (OSN). To establish this, an OSN-assisted caching strategy is proposed which is evaluated using a large Twitter dataset and through simulation. Results show that vanilla LRU mostly outperforms the OSN-aided mechanisms.

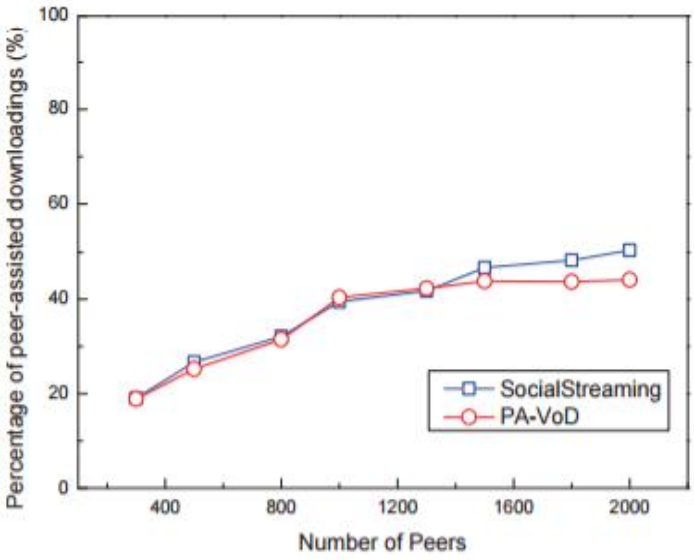


Figure 33 - Percentage of peer-assisted downloadings in SocialStreaming and PA-VoD

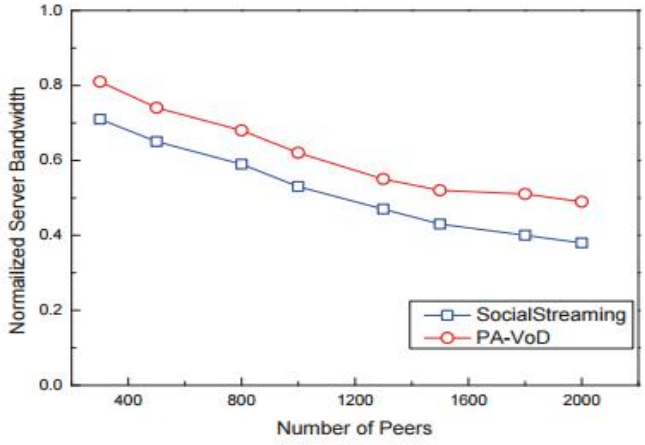


Figure 34 - Server bandwidth consumptions of SocialStreaming and PAVoD

3 ENERGY-EFFICIENT CONTENT MANAGEMENT IN CONTENT DELIVERY NETWORKS (CDNS)

In this section, we present our contributions to the design of energy-efficient CDNs. First, we investigate the anatomy of end-to-end energy usage for the delivery of video content and compare the implication of the video coding scheme on the end-to-end delivery of video content (Section 3.1). Then, in Section 3.1 we present a proposed solution for optimal latency and energy efficient content management in CDNs. In Section 3.2, an energy efficient in-network caching mechanism for CCN is presented. Section 3.3 describes a web-based, resource-aware, and distributed video delivery architecture for mobile environments. Finally, in Section 3.3 a video delivery scheme relying on social information of users and their resources (i.e. processing and storage of user devices) is presented

3.1 Anatomy of End-to-End Energy Usage for Video Delivery and its Implications for Video Content Generation

3.1.1 Introduction

Energy efficient computing has attracted significant attention from various researchers from different fields. While previous efforts have offered valuable insights into particular elements of energy consumption, only a few studies such as [44] [32] have addressed the problem from an end-to-end perspective. However, these studies treat the content in a simplistic way (i.e. streams of bits and packets carried over a network path), and do not consider the energy used for processing (e.g. encoding) the content at both source and destination where the content is released and pre-processed before is consumed. The energy usage when processing content makes a significant difference in the overall end-to-end energy usage behavior of a given content. Therefore, the amount of energy used to encode (in the source) and decode at the user device must not be ignored. Putting this in the context of the most popular video coding schemes, video content encoded with H.265/HEVC have a much higher compression ratio than H.264/AVC [45]. This leads to higher energy saving for the transmission of H.265/HEVC encoded video content. However, the energy saving gain of H.265/HEVC due to its high compression comes at the cost of higher processing complexity (an thus larger energy usage) compared to H.264/AVC [45]. Therefore, the two coding schemes affect the total energy usage of video delivery from an end-to-end standpoint. It is also as important to take into account the popularity (i.e. the relative demand rate) of content. The question is then which encoding format to use for a given popularity in order to save energy.

Driven by these motivations, in this work, we propose an end-to-end model of energy usage and apply this to two use cases; using H.264/AVC as the most prevalent codec at the time of writing, and using H.265/HEVC which is expected to gain a high popularity in the digital society. For a sample video, we measure its encoding and decoding energy usage and apply these parameters to the proposed model in the two use cases. We also investigate the impact of video popularity on the energy usage and determine for which combination of video sequence length and popularity levels the video is ought to be encoded and delivered in a given format. Then, we consider a hybrid scheme where contents format determined according to their predicted popularity. Throughout this study the quality of service (QoS) of video (measured in Peak Signal to Noise Ratio (PSNR)) is fixed for all scenarios of video encoding and across all encoded presets of the sample video. Although our results apply only to a single video sample used in our measurements, the proposed approach is sufficiently general to be applied to a variety of video samples with different attributes. Throughout this study, we use the terms H.265 and HEVC interchangeably.

Our model includes the energy usage from three components, first video processing, i.e., both encoding and decoding, second video content delivery from the server to the network edge and last wireless links from the network edge to the end devices. Video processing is one of the most energy intensive parts of a video delivery system. Some works (e.g., in [46]) investigate the energy usage of codecs on different platforms. Some works focus on system design aspects, e.g. in [47]. In [48] the authors concentrated on improving the energy efficiency in codec design. In some works there is a trend to design a software-based encoder e.g in [49]. The most straightforward method which relates the complexity to energy usage is discussed in [50]. There are also some works on comparing the complexity of H.265/HEVC and H.264/AVC codecs in [45] which we find match the results we obtain in this work from an energy perspective. The energy usage on video content delivery is another important part. In [44] [12] the authors proposed an end-to-end energy usage model for video delivery. [32] focused on architecture and proposed two models for CDN and CCN for video content,

respectively. However, as far as we know, none of the previous works considers different video formats and their decoding/encoding issues within a CDN. Mobile devices have a significant effect on overall energy usage. In [51] a power consumption model was proposed according to frame rate on WiFi 802.11 devices. In [52] a detailed model for energy consumption in 802.11ac access points was presented. In our work, we assume that users have wireless devices.
The remainder of this section is organized as follows: Section 3.1.2 describes our proposed end-to-end energy consumption model. Our numerical results are presented in Section 3.1.3, and the study is concluded in Section 3.1.4.

3.1.2 Proposed End to End Energy Usage Model

We consider the case where there is video content with a given popularity, and compare the end-to-end energy usage for the delivery of such content in our two use cases. We conduct this study within the context of a content delivery framework, where our content is situated within a pool of contents in the system ranked by their popularities. Denote by a set $\mathrm{K} = \{1,2,..., K\}$ the pool of video contents, where k represents the rank of the content of interest. Contents with lower ranks have higher popularity. The total demand for all content is R and the demand for content k, denoted by $R(k)$, is obtained using a Zipf distribution, i.e. $R(k) = R\frac{k^{-\alpha}}{\sum_{k=1}^{K} k^{-\alpha}}$, where α is the Zipfian exponent. More specifically, $R(k)$ is the total demand for the content k during its lifetime, i.e. from the time it is generated until the time it is eliminated from the system. By producing new content, we mean the encoding of a given content into a set of versions $\mathrm{V} = \{1,2,..., V\}$ and retaining all versions in the system, in a fashion similar to how the major content providers such as YouTube generate their video content. The set V is similar in the two encoding schemes H.264 and HEVC. More concretely, this set consists of 10 different presets, ranging from a very fast to a very slow preset. The size of a version v of content k is $s_{h4}(k,v)$ bits if it is encoded with H.264, and $s_{h5}(k,v)$ when encoded with H.265. The indexes are dropped in those expressions applied generally to both codecs. When a demand for content k arrives, it is served with a version v with probability $p(v)$, where p is a probability measure over the entire set V. This probability distribution is independent of the codec type used for encoding the content. Such a distribution is realized in practice by the available bandwidth, or by device type and the streaming protocol e.g. HTTP DASH. For simplification, we abstract away such details by using a general distribution p. We mark that, each version v could also be produced into additional versions differing with respect to the video bitrate, however, we do not consider this case in our study because the presets in set V cover this property; i.e. the bitrates of presets are usually different.
With the aforementioned content delivery framework, we approach the energy usage from an end-to-end point of view, taking into account the total energy used at the source where the content is encoded, transmitted through a core and edge network, and to a WiFi access point (AP) that delivers the content to an associated user device which finally receives and decodes the content. Our reason for choosing WiFi technology as the representative access network in this work is two-fold: first, it is already regarded as a prevalent access technology, and will continue to grow even more rapidly due to the need for offloading a significant portion of cellular mobile traffic (in 2015 around 55% [53]]), thanks to the penetration of dual-model devices. Second, we found that the existing models for WiFi energy usage are by far more mature compared to cellular access alternatives such as LTE. We serve all demands of a content k as follows,

$$\begin{aligned}\mathbf{E}(\xi(k)) = \xi_e(k) + \xi_g(k) + \sum_{v=1}^{V} p(v)R(k)(\xi_{tx}(k,v) \\ + \xi_s(k,v) + \xi_a(k,v) + \xi_r(k,v) + \xi_d(k,v))\end{aligned} \quad (1)$$

where the contributions from individual components are expressed as a function of k and v to account for the general attributes of the content such as its demand, and the particular attributes related to individual versions of the content, notably the version size in number of bits. In (1), $\xi_e(k)$ is

the energy consumed to encode all presets of content k, $\xi_g(k)$ is the storage energy used to retain all versions of k, $\xi_{tx}(k,v)$ is the transmission (core and edge) energy used for version v, $\xi_s(k,v)$ is the streaming energy usage, $\xi_a(k,v)$ is the energy used in the access point to transmit the packet streams and to receive acknowledgement (ACK), $\xi_r(k,v)$ and $\xi_d(k,v)$ are the energies used to receive and decode the content, respectively. We do not account for energy used for video playout since it is highly dependent on device screen size, brightness, etc.

In the following, the individual components of the expected energy usage expressed by (1) are specified in more detail.

Transmission, storage and streaming: we extend the model proposed in [32] for conventional CDNs to calculate the energy usage caused by transmission in the core, metro and edge networks, as well as storage and the server. In this model, it is assumed that content residing on a content provider server traverses a LAN switch and an edge router in the content provider data center, then passes H routers in the core, nodes in the metro and edge network including a provider edge router, two switches and a gateway router. This is a typical configuration of an IPTV network [44]. We note that the parameter H is defined as the number of hops traversed in the core network using a shortest path routing policy. The transmission energy (in Joules), corresponding to a version v of content k, is expressed as:

$$\begin{aligned}\xi_{tx}(k,v) = 4\delta\, s(k,v)[(e_d^r + e_d^{oxc})(H+1) \\ + e_d^{wdm} H] + 4\delta\, s(k,v)(3e_d^e + e_d^g + 2e_d^{pe})\end{aligned} \tag{2}$$

where the first term describes the energy usage in the core, and the second term is the energy usage caused by the transmitting nodes in the provider edge network and in the data center. e_d^r, e_d^{oxc} and e_d^{wdm} denote the energy density (J/bit) of the core router, optical cross connect, and WDM links respectively. e_d^e, e_d^g and e_d^{pe} represent the energy densities of an Ethernet switch, gateway router, and provider edge router respectively (J/bit). The factor 4 in expression (2) accounts for redundancy and overhead [44]. We have introduced a new parameter $\delta > 1$ into the model in [32] to account for the overhead incurred by streaming protocol and packetization (TCP/IP headers). We note that the model in [32] assumes the access network is a passive optical network whose energy usage does not depend on traffic load, and therefore the energy usage in the access network is omitted. We apply the same assumption in this work but we assume the passive optical network (PON) is extended with wireless access comprising a WiFi access point and a user device, which in contrast to PON, uses energy according to the offered load [51].

The energy usage of the streaming server is expressed as follows [32],

$$\xi_s(k,v) = s(k,v) e_d^s \tag{3}$$

where e_d^s is the energy density of the server (J/bit). For the storage we have [32],

$$\xi_g(k) = \sum_{v=1}^{V} s(k,v)\, p_d^g T \tag{4}$$

where p_d^g is the power utilization (in W/bit) of the storage device and T is the residence time duration of the content on the storage. We assume this time duration is the same for all contents.

WiFi Access: in the following, we characterize the power usage in the access point (ξ_a) and the device (ξ_r) for transmitting and receiving the content. We apply a model recently developed in [51] and customize it for the purpose of our study. In particular, the model expressed by Eq. 8 in [51] describes the power usage of the device only. We extend this model to account for both access point and device. The power usage in the device is described as:

$$\xi_r(k,v) = [\rho_{id}^r + \lambda(k,v)(\rho_{tx}^r T_{ack} + \rho_{rx}^r T_L + \gamma_{xr}^r)]T(k) \tag{5}$$

where ρ_{id}^{r} is the idle power of the device, $\lambda(k,v)$ is the frame rate (seen from the device MAC layer), ρ_{tx}^{r} is the transmission power, ρ_{rx}^{r} is the receive power (possibly different to the transmission power), T_{ack} is the airtime of an ACK packet which includes the MAC and physical header, and depends on the wireless bitrate. T_L is the airtime of the video packet (including physical and MAC headers). γ_{xr}^{r} is the cross factor power, used to process the packet throughout the entire protocol stack of the device. All aforementioned power related parameters are measured in Watt. The parameter T_k –the duration of the video– is introduced in (5) to obtain the equivalent energy usage measured in Joule. For the access point, the expression is:

$$\xi_a(k,v) = [\rho_{id}^{a} + \lambda(k,v)(\rho_{tx}^{a} T_L(1+R_{tx}) + \rho_{rx}^{a} T_{ack} + \gamma_{xg}^{a})]T(k) \quad (6)$$

where R_{tx} is the number of frame re-transmissions caused by failed receptions in the device. This is a MAC layer parameter which depends on the traffic load and density of nodes associated with the AP. In our numerical analysis in Section 3, we will use a range of predetermined re-transmission rates to understand its effect on the energy usage in the WiFi access. γ_{xg}^{a} is the cross-factor power of the AP. The meaning of the remaining parameters in (6) follow the same notation as for the device (5)

Encoding: we measure the encoding energy usage of a sample video separately for H.264 MPEG 4/AVC and H.265 MPEG-H/HEVC. This measurement involves encoding a single raw video file (YUV format) into an mp4 container. The parameters of the video file are demonstrated in Table 8.

Table 8: Attributes of the sample video

Video Parameter	Value
Length	43s 867ms
Frame rate	30 fps
Width	1920 pixels
Height	1080 pixels
Color space	YUV
Chroma subsampling	4:2:0

The encoding is performed on a Dell PowerEdge R150 rack server with two Xeon E5606 CPUs. We use open source codec software x264 and x265 for encoding the video to H.264 and HEVC formats respectively. We used an Eaton ePDU device to measure the power draw. Among multiple measurements we have conducted with different configurations, we measured with fixed PSNR across all presets of the video. The encoding energies are listed in Table 9. Recall that $\xi_e(k)$ in (1) represents the encoding energy. In our sample video, $\xi_e(k)$ of a codec is the sum of energy usage for all presets of that codec.

Table 9: Encoding energy consumption and video bitrate per codec per video preset

	H.264/AVC		H.265/HEVC	
Presets	bitrate (Kb/s)	Encoding Energy(J)	bitrate (Kb/s)	Encoding Energy(J)
ultrafast	6799.41	685	853	4650
superfast	3361.31	1080	806	5660
veryfast	1929.98	2340	805	9650

faster	2298.39	3420	735	10900
fast	2242.38	4510	612	13500
medium	2098.08	5030	579	18600
slow	2002.02	9740	547	51700
slower	1866.73	15900	515	117000
veryslow	1696.64	33400	519	179000
placebo	1725.28	131000	519	277000

We observe that the encoding energy of HEVC is multiple times higher than the encoding energy of H.264. This conforms to previous studies that studied the encoding complexity of H.264 and HEVC [45].

Decoding: we measured the energy usage of decoding on an Apple iPad Air 2 tablet, separately for the sample video that we encoded with H.264 and HEVC. We use the VLC Player to decode all presets of H.264 video, and HEVCDecoder with 4 threads to decode the presets of the HEVC encoded video. The reason for choosing a multi-thread configuration for HEVCDecoder is that the HEVC codec is fundamentally designed to support this feature. For power measurements, we used the Energy Diagnostics profiler from Apple Instruments software. The energy draw values along with video preset sizes in byte are listed in Table 10.

Table 10: Decoding energy and byte size per codec and video preset

	H.264/AVC		H.265/HEVC	
Presets	Decoding Energy(J)	Video size (B)	Decoding Energy(J)	Video size (B)
ultrafast	48.09096	37283715	62.54359123	4677318
superfast	56.81238	18431323	65.53379071	4419600
veryfast	57.31074	10582804	72.01255625	4414116
faster	74.75357	12602934	65.28460742	4030280
fast	67.27807	12295810	63.04195781	3355825
medium	60.05176	11504559	63.04195781	3174874
slow	58.05829	10977826	63.54032439	2999406
slower	58.55666	10235981	70.51745651	2823938
veryslow	58.30748	9303313	67.02889045	2845871
placebo	63.29114	9460357	66.28134058	2845871

We observe that the energy usage of HEVC decoding is larger than those of H.264 for most presets, although the difference is not as significant as it is when encoding. An study conducted in [45] also confirms our observation, although their approach is based on the time complexity of decoding. The encoding and decoding energy altogether might draw us to conclude that H.264 is more energy efficient than HEVC. However, as we will show in our simulation study in Section 3.1.3, it is not always the case when end-to-end energy usage is considered.

3.1.3 Simulation Results

Simulation Setup

A video stream is encoded at an origin server, delivered to an end-device via a WiFi access point, and decoded by the device. There are ten different encoding presets for each video format from placebo to ultrafast. We use the same sample video as shown in Section 3.1.2 and its attributes and encoding and decoding related values listed in Table 8, Table 9 and Table 10, respectively. For the most part of our simulations, we set the exponent of the Zipfian distribution to 1.2 as suggested in [54] (see Table 11 for the remaining parameters). The energy usage per bit for networking devices {storage, server,

core/edge/gateway routers, WDM link, OXC and Ethernet switches} follow those in [32]. The idle power, cross-factors, and reception/transmission power coefficients of the access point and device are set according to [51]. More specifically, for the device, we choose the power coefficients corresponding to Samsung Galaxy Note 10.1 from [51] because this device is the closest to the device we used for the measurement of decoding energy.Also, the access point parameters are the same as the Linksys WRT54GL device that is used in [51]. We assume the access point and device are in non-sleep mode in order to receive data and decode the video. We also take into account the idle power of the access point and device and use UDP video streaming.

Table 11: Simulation parameters

parameter	value
Zipf Exponent	0.5, 1.2, 2.5
Number of videos	100
Total demands	100000
Raw video packet length	1400B
Percentage packet overhead (δ)	1.03
Device idle power	1.9935 W
Number of core hops (H)	variable from 0-12
Content lifetime	90 days

Composition of the End-to-End Energy Usage

In this subsection, we consider the storage, server, transmission, wireless, decoding and idle energy without taking encoding energy into account, because a video is normally encoded once but viewed several times. However, we address the effects of retransmissions in the access point and the number of hops in the core. The former reflects the WiFi network density, while the latter describes the distance from the content provider server to the consumer. We mark that, in this subsection, the number of demands for our sample video is obtained using a Zipf distribution with exponent 1.2 and with a total demand shown in Table 11.
First, we study how WiFi retransmissions influence the energy usage. We fix the number of hops in the core network to five. Figure 35 shows the fraction of energy used by the individual parts of the network to deliver the video content with rank 1, i.e., the most popular video. In the figure, the number of retransmission in the access point is set to 0 (ideal case). Also, for the sake of a fine granular comparison, we separate the devices' idle energy from the energy used for the WiFi interface. By WiFi energy, we mean the energy used in both the access point and device for transmitting and receiving packets and processing them in the protocol stack. From the figure, we observe that the decoding energy percentage is lower in H.264, although its transmission energy is higher compared to HEVC. The idle states of the access point and the end user device use a substantially large fraction of the total energy. Also, we observe that in both H.264 and HEVC the decoding energy is much higher compared to the energy used for WiFi transmission and reception.

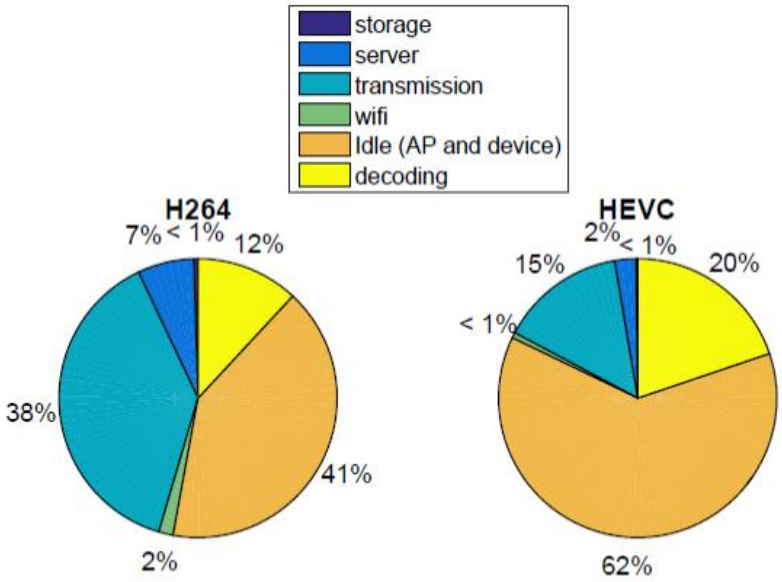


Figure 35: The contributions of individual parts to energy consumption

We increase the number of retransmissions and investigate the effects on WiFi energy usage in Figure 36. Recall that when the device downloads the video, it only transmits ACK packets, while the access point transmits video packets and receives ACKs from the user device. Since the device only acks a successful transmission, retransmissions only affect the energy of the access point. We observe that the slope of the H.264 curve is steeper than HEVC due to its larger traffic volume (i.e. larger video size). We study retransmission effects when the number of retransmissions is maximum (i.e. five), and show the results in Figure 37. Compared to Figure 35, the WiFi part of the total energy increases, but is still only a small fraction. This implies that the WiFi energy usage, even under very dense network conditions, is negligible compared to decoding and device idle usage. Thus, future research should focus on optimizing the energy efficiency of video decoding and the device idle state.

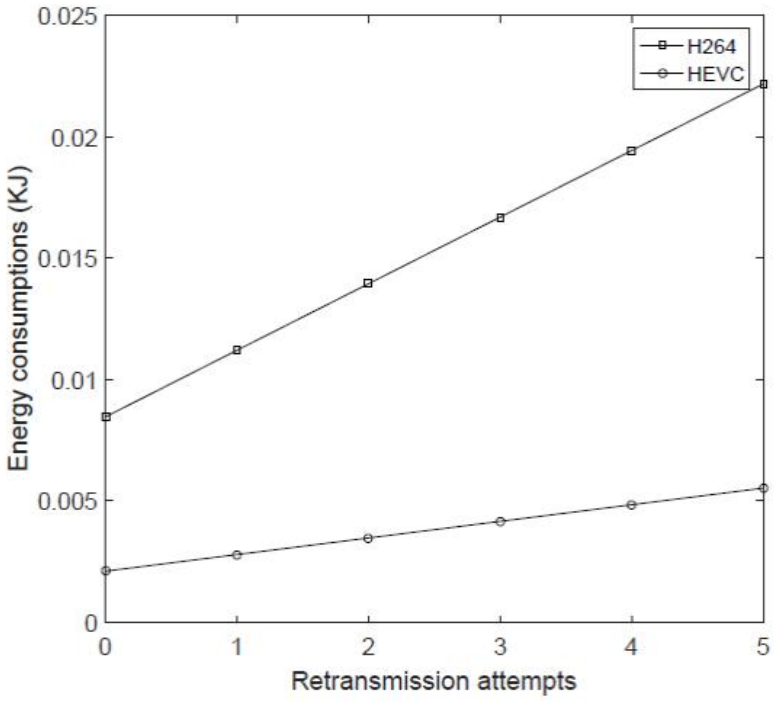


Figure 36: The effect of frame retransmission on WiFi energy usage

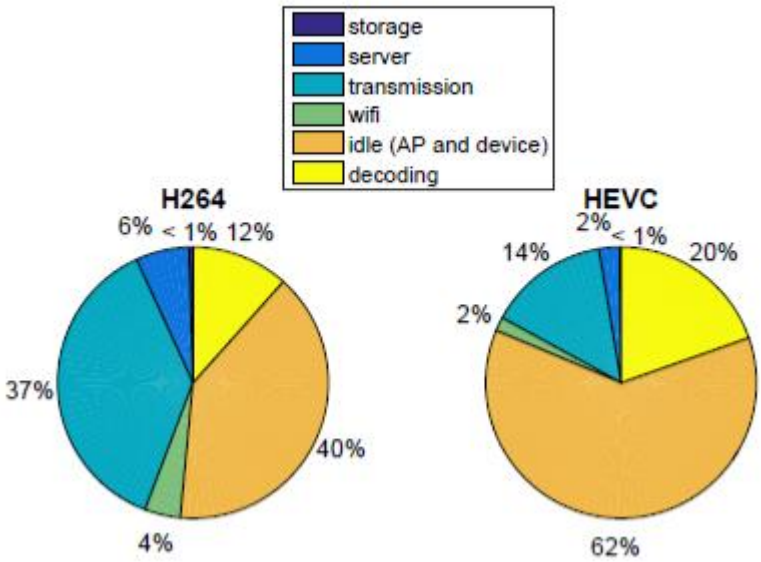


Figure 37: Energy usage with the highest number of frame retransmissions

We also investigate how the number of hops in the core network affects the results. Figure 38 shows the transmission energy as a function of the number of hops traversed by a single request of the sample video. We observe that the slope of the H.264 energy curve is again higher than that of HEVC due to its larger traffic size. For H.264, when the number of hops increases from 0 to 5, the transmission energy grows by 73%; whereas for HEVC, the increase is only 33% from 0 to 5 hops. Therefore, in order to save energy when distributing H.264 content, it is important to store the content closer to the network edge as the transmission energy for H.264 accounts for more than one third of the total energy as shown in Figure 35 and Figure 37.

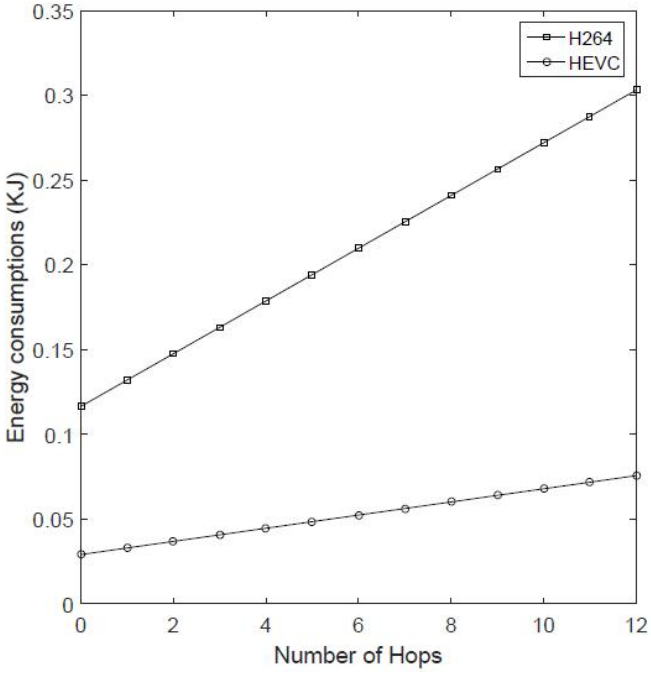


Figure 38: The impact of number of hops on transmission energy

Analysis of Energy with Respect to Video Popularity

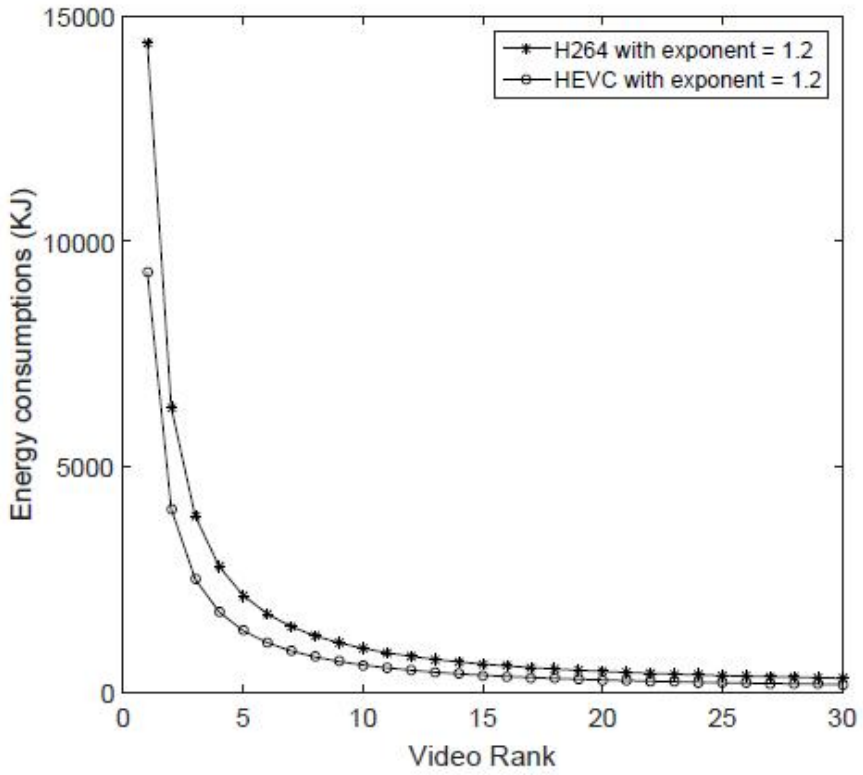


Figure 39: Video delivery energy usage as a function of video rank (encoding energy not included)

In this section, we compare the energy usage for different video ranks. The number of hops in the core network is set to 5 and the number of retransmissions over WiFi is set to its maximum value 5. We use a Zipf distribution with exponent 1.2 to determine the demand rate corresponding to each rank of the video sample. Figure 39 shows the energy usage corresponding to each video rank, taking into account all factors except for encoding. We observe that, for all video ranks, the HEVC energy usage is lower than H.264 because HEVC saves substantially in the transmission, and at the same time its decoding energy is marginally higher than H.264 (around 15% in the sample video). However, when encoding energy is taken into consideration, the situation becomes radically different.

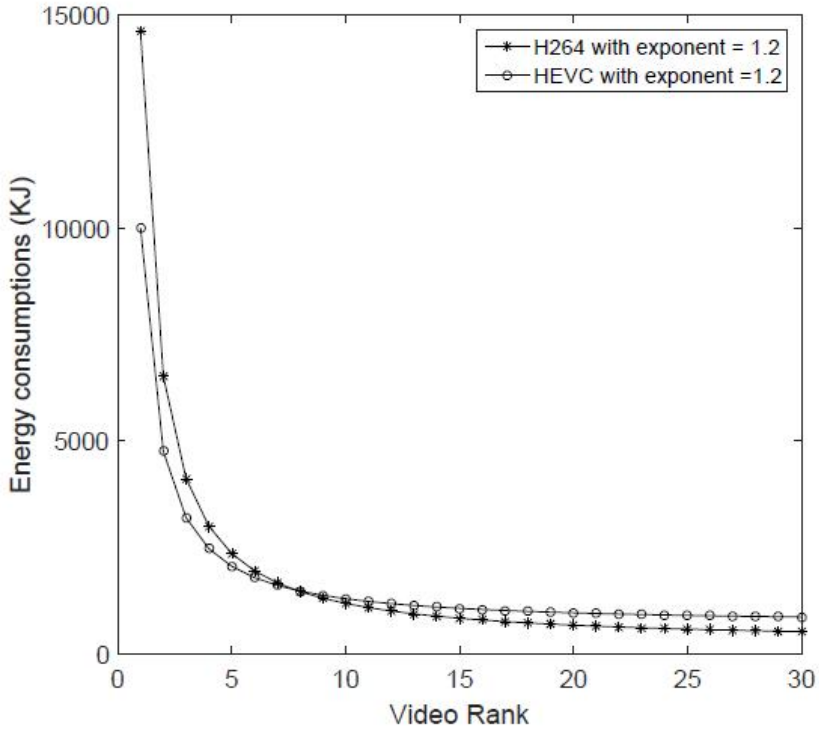


Figure 40: The end-to-end video delivery energy as a function of video rank (encoding energy is included)

Figure 40 shows that when encoding is included, the H.264 end-to-end energy is lower than HEVC when the video rank exceeds 8. It means that if we have a video content which is estimated to be downloaded with a low rate, e.g., in many video transmission applications like video chats and video conferences or user-generated video clips, it is more energy efficient to encode and deliver it in using H.264. Conversely, HEVC is more energy efficient than H.264 when the popularity of a video is high. This phenomenon is attributed to the substantially large difference in energy required to encode video using HEVC and H.264 as shown in Table 9 (HEVC encoding energy is $>$ 3 times larger than H.264). This implies that there must be a sufficiently large number of requests per video in order for HEVC to become more energy efficient than H.264. We note that, in our simulations with Zipf exponent of 1.2, the crossing point (i.e. the initial video rank where H.264 becomes more efficient) coincides with video rank 8 which accounts for about 2500 requests, and many videos in real situations may not have such a demand rate.

As mentioned above, there is a crossing point between the two curves in Figure 40. We regard this crossing point as a threshold. Before the threshold, i.e., for more popular videos, the end-to-end energy for each video rank is lower for HEVC, whereas above the threshold, H.264 uses less energy. We conduct simulations with different Zipf exponents to study how the shape of popularity distribution affects the global energy usage (i.e. all content, collectively) and the energy usage for the subset of contents falling below the threshold and the subset which exceeds the threshold. Figure 41 compares the absolute global energy values and percentages of energy for the subsets below and above the threshold. With larger exponents, the energy usage of popular videos becomes higher. The global energy usage for HEVC is larger than H.264 because of the long tail property of the Zipfian distribution, i.e., the majority of videos falls below the threshold when HEVC becomes more energy efficient. This observation implies to use H.264 for content that is estimated to have a limited number of downloads and use HEVC for highly popular content (e.g. News videos or professional video content) in order to save energy. We denote this policy *hybrid* in contrast to a H.264 or HEVC only policy. From Figure 41, we see that the hybrid scheme is the most energy efficient. For our sample video with our analysis assumptions, if we use the hybrid policy with Zipf exponent 1.2, we can save 14% energy compared to a H.264 only policy and save 29% energy compared to a HEVC only policy. Moreover, the hybrid policy becomes more energy efficient with higher Zipf exponents.

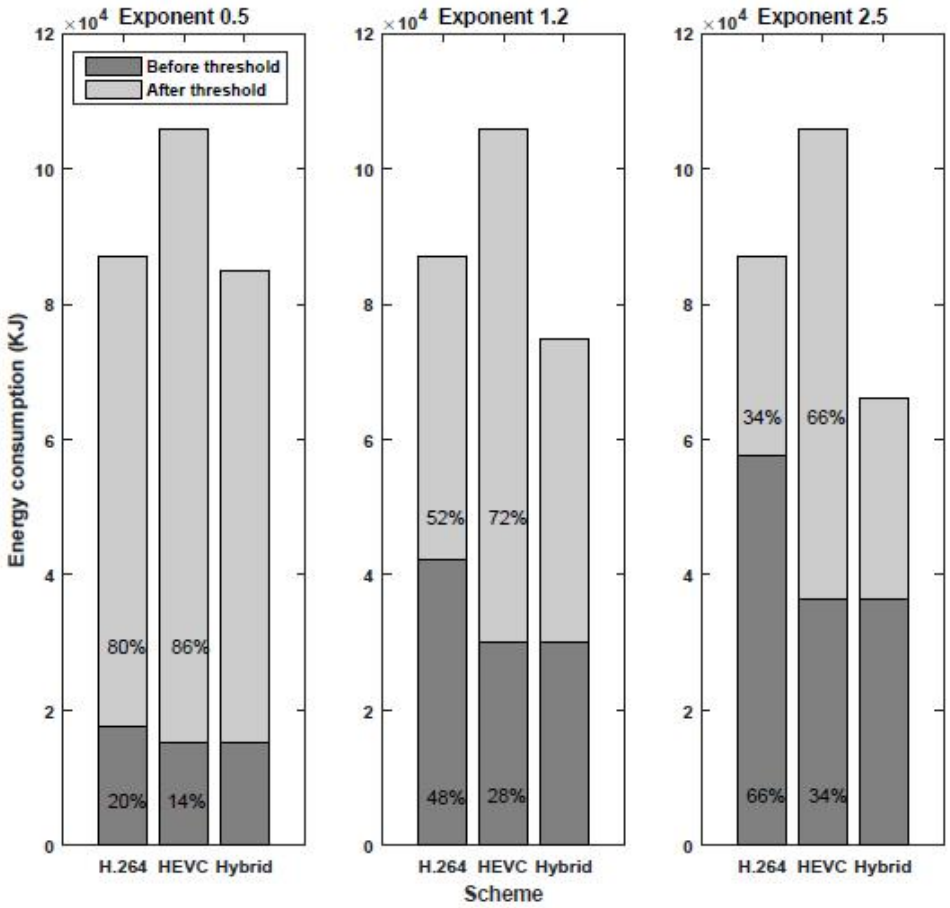


Figure 41: The global energy of all requests with respect to the threshold

Device Energy Usage Analysis

End users are usually only concerned with energy usage in order to increase battery life time. In this subsection, we only consider the energy usage of WiFi, video decoding and idle time for the device in order to capture this aspect. The H.264 decoding energy is a bit lower than HEVC on average as shown in Table 10. However, for receiving traffic via WiFi and processing it in the protocol stack, HEVC uses less energy because of the smaller traffic size. Figure 42 compares the energy of H.264 and HEVC on the end device for different presets. HEVC uses less energy for the "fast" and "faster" presets but more for the remaining presets of the sample video. On average, H.264 and HEVC use a similar amount of energy. Their difference is only 3% and the device idle power dominates. Therefore, from the end-user perspective, the choice between HEVC and H.264 is unimportant.

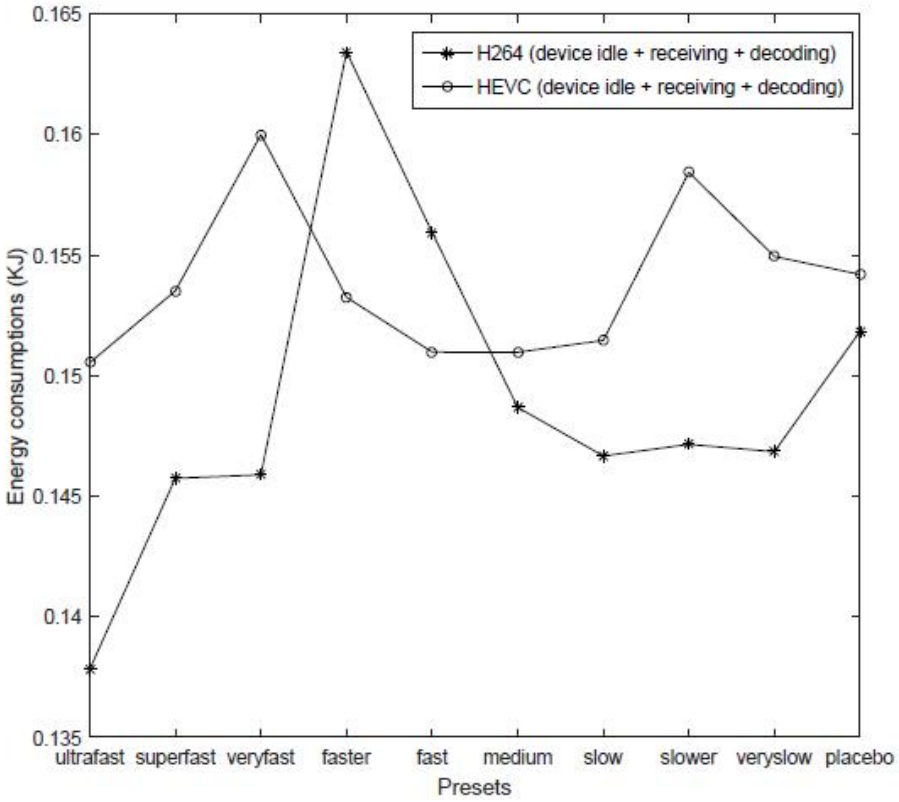


Figure 42: Energy consumption in the user device

Energy Analysis of Different Video Sizes

So far, our analysis has been based on a single tested video. For completeness, we here account for the effect of varying video length on the energy usage. To construct a model for the energy usage due to varying video length we assume that the complexity of the video is independent on its length so the energy usage for encoding and decoding a video is linearly dependent on the length. We denote a precoded version of a video using a specific preset by k, where k contains the characteristics of the specific version, i.e. $k=(s_{pre}(k),s(k),...)$ which all depend on the size of the video $s_{pre}(k)$. The encoding energy is then denoted $\xi_e(k)$. Because of the linearity we then introduce an enlargement factor n derived as $\xi_e(ns_{pre}(k),s(k),...)=n\xi_e(s_{pre}(k),s(k),...)$ where the same holds true for $\xi_g(k)$. By considering relations (2)(3)(4) we see that the function is linear and mapping on $s(k,v)$ is possible because we have no constant factor. Relations (5)(6) also linearly depend on the video length as $T(k)$ linearly depends on video size. In order to highlight the effect of this, we plot the end-to-end energy usage of three video sizes, base size, 10 times base size and 20 times base size, against the popularity of the video.

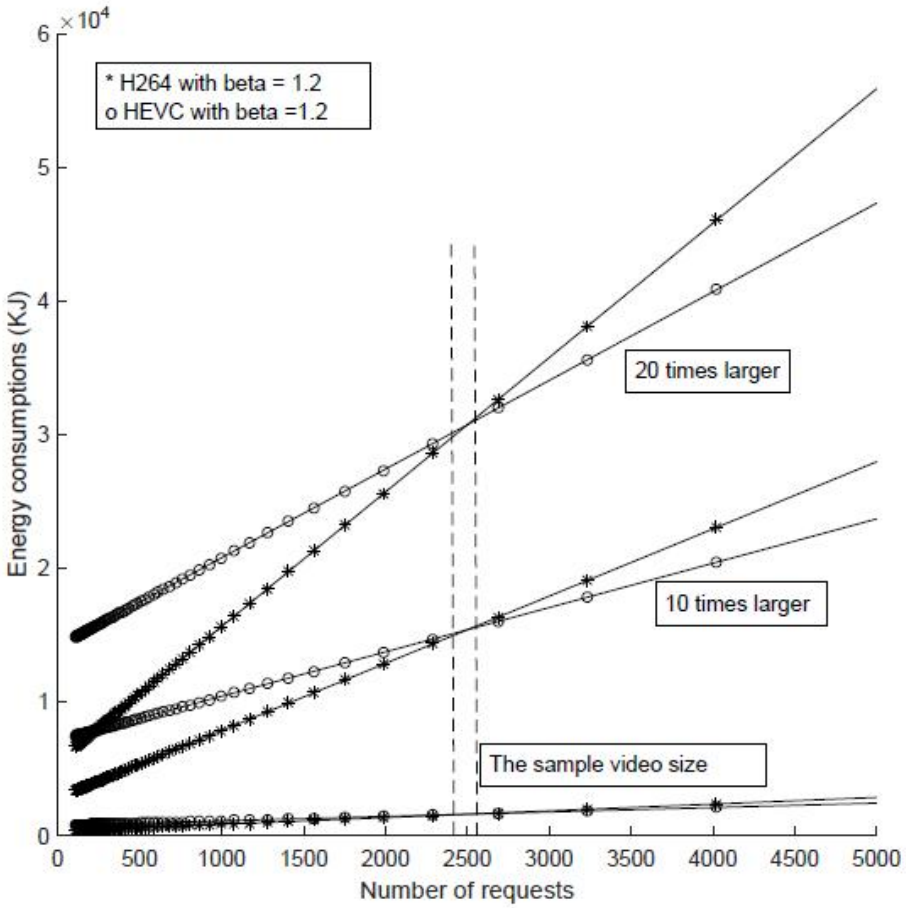


Figure 43: Energy consumption with different video sizes

Figure 43 shows that the crossing points (thresholds) do not change when the video size changes, only the absolute energy usage increases with size. Furthermore, the difference in energy usage between the different codecs increases with the file size, meaning that there are bigger energy gains to be had as the video file sizes increase.

3.1.4 Conclusion

We propose an end-to-end energy model and use it in two scenarios, differentiated by the coding format of the video content. For a sample video encoded with H.264/AVC and H.265/HEVC, we build its energy profile based on measurements conducted separately for encoding and decoding. We find that, neither coding scheme always lead to the lowest energy usage and the best choice is dependant on the popularity of the video. The results also have implications for future research as we showed that the WiFi energy component is negligible compared to the decoding component. Also encoding drains an amount of energy much larger than both the WiFi transfer and the decoding. The concern escalates for battery-operated devices. Therefore, it is important for future research to focus more on the encoding and decoding aspects. Second, the idle energy draw of a device is multiple orders of magnitude larger than the WiFi interface for traffic transmission and reception. This implies that it is important that future research focus on lowering the base power of user device.

3.2 Collaborative Content Replication and Request Routing

3.2.1 Introduction

The proliferation of content delivery networks as the present solution to content transfer worldwide has led to bandwidth saving and shorter content access delay experienced by end-users [55] [56]. A typical CDN such as Akamai is built as an overlay network comprising load balancing and request routing server(s) and many regional content distribution elements known as surrogate or delivery servers. Content delivery is realized by means of two intertwined major tasks known as content replication and request routing (CRRR) [55]. The replication of content involves caching the content in response to the change in the content demand and/or change in the state of the content distribution network. Request routing performs assignment of user requests to some content delivery server(s) with the closest proximity to the user's location, where the proximity is defined by various static factors such as physical distance and dynamic factors such as communication delay and server response time [57]. CRRR aims to enhance user satisfaction and, at the same time, minimize the cost incurred by content providers and distributors.
Unarguably, the physical characteristics of a CDN such as node placement, density, and processing power of servers and the installed storage capacity per server highly influence the degree to which the CDN's promised goals can be achieved. The ultimate performance of a CDN is also driven by the efficiency of CRRR tasks. Thus, with a limited number and quality of infrastructure nodes and given an optimal placement of those nodes, further improvement of efficiency can merely be sought through the design and implementation of optimal CRRR functions. A traditional approach to content replication is either full or partial replication coupled with non-cooperative pull-based caching where each delivery node, upon receiving a content request, decides to cache the content. The cache update process, i.e., the decisions on replicating the new content and the eviction (i.e., purging) of old content(s), is performed locally by the delivery node and using some locally measured indicator(s) such as relative frequency, recency, or popularity of contents subject to caching or eviction. Cooperative pull-based content replication and request routing (C ^{2}R ^{3}) [55], as an alternative approach in content delivery, have proven to be a promising mechanism to gain better efficiency in CDNs. The existing approaches to C ^{2}R ^{3} in the CDN context exhibit a number of shortcomings including restriction to proprietary CDN structures (e.g., tree shape and hierarchical) and excessive focus on bandwidth saving rather than content access delay. Other approaches to C ^{2}R ^{3} can be found in the context of web caching. This class of techniques was originally designed to improve hit rate as the main objective. It is arguable whether these methods improve access delay because an optimal hit rate does not necessarily translate to optimal access delay [58]. In this work, we address a general CDN architecture and formulate C ^{2}R ^{3} as an optimization problem with the objective of delay minimization. The optimal C ^{2}R ^{3} obtained from the optimization problem is used as a benchmark to investigate the performance of two distributed popularity-based algorithms we propose for C ^{2}R ^{3}. Also, we investigate the impacts of the key content and network related parameters such as popularity distribution and cache size of servers on the performance of proposed methods. As another contribution, we compare C ^{2}R ^{3} to a cooperative recency-based method representing traditional web caching techniques. While the delay improvements is the main objective of this work, as we will show later, the proposed solutions to content replication and request routing also demonstrate a signifcant power saving in the transmission network.

3.2.2 Problem description and model

We consider a general CDN topology comprising N regional content servers (RCS) and M main content servers (MCS). Hereafter, the set of RCSs and MCSs will be denoted by RS and MS, respectively. The set RS forms a complete mesh, while the servers in MS do not interact at all. Each RCS in RS forms a bipartite graph with the servers in set MS. There is a content pool represented by a set C of K content objects each located on at most one MCS. We represent the predefined locations of content objects by a matrix $Y = \{y_{ci} | \forall c \in C \ \& \ \forall i \in MS\}$, where $y_{ci} = 1$ if c is located in MCS_i otherwise $y_{ci} = 0$. A content $c \in C$ has a size s_c defined as an integer multiple of content unit size or *chunk*, for simplicity. The demand rate of c in RCS_i is d_{ci} and all requests for c issued by users in the region of RCS_i are initially directed to this RCS_i. The cache (or storage) capacity of RCS_i is B_i, defined as the maximum number of content chunks which can be accommodated in the RCS. Upon a request received in a RCS_i for content c, it serves the request if a copy of the content is found in the local cache, otherwise it redirects the request to a peer RCS or to a MCS, depending on the routing policy which, in turn, is driven by the inter-server transmission and the internal processing delay of servers.

We assume the transmission between the servers of the CDN accounts for the first source of content access delay. Accordingly, we represent by τ_{ij} and δ_{im} the transmission delay involved in the access of a content chunk from RCS_j to RCS_i and from MCS_m to RCS_i, respectively. The processing and queuing delay of the servers accounts for the second source of delay experienced by the end user. This type of delay is obtained using a suitable function $\phi_j(Z_j)$ of the load incurred to server j. Generally, the load of a RCS_j can be defined by a set of triplets $Z_j \doteq \{(x_{cij}, d_{ci}, s_c) | \forall c \in C\ \forall i \in RS\}$, where x_{cij} is a binary variable indicating whether RCS_i redirects the requests for content c to RCS_j. d_{ci} is the demand rate for content c in RCS_i and s_c is the size of the content. Similarly, the load of a MCS_j is characterized by a set of triplets $Z_j \doteq \{(y_{cij}, d_{ci}, s_c) | \forall c \in C\ \forall i \in RS\}$, where y_{cij} has the same meaning of x_{cij} but is used to indicate request routing from RCSs to MCS_j. We assume the delay between the end-user device and the associated regional server is negligible. With this setting in mind, the optimal C^{2}R^3 policy is determined by solving the following Integer Programming (IP) problem:

$$\text{Minimize} \quad \sum_{c=1}^{K}\sum_{i=1}^{N} s_c d_{ci}\left(\sum_{j=1}^{N}\tau_{ij}x_{cij} + \sum_{j=1}^{M}\delta_{ij}y_{cij}\right) + \sum_{i=1}^{N+M}\phi_i(Z_i) \tag{1}$$

$$\text{Subject to:} \quad x_{cj} - x_{cij} \geq 0, \qquad \forall c \in C\ \ i,j \in RS \tag{2}$$

$$y_{cj} - y_{cij} \geq 0, \qquad \forall c \in C\ \ i \in RS\ \ j \in MS \tag{3}$$

$$\sum_{c=1}^{K} s_c x_{ci} \leq B_i, \quad \forall i \in RS \tag{4}$$

$$\sum_{j=1}^{N} x_{cij} + \sum_{j=1}^{M} y_{cij} = 1, \qquad \forall c \in C\ \ i \in RS \tag{5}$$

$$x_{cj}, \quad x_{cij}, \quad y_{cj}, \quad y_{cij} \in \{0,1\} \tag{6}$$

x_{cj} is the replication binary variable indicating whether a content c should be replicated in RCS_j. y_{cj} is a predetermined binary constant indicating whether content c is located in MCS_j. The set of binary variables $\{x_{cj}\}$, $\{x_{cij}\}$, and $\{y_{cij}\}$ characterize the C^{2}R^3 solution obtained from (1). Constraint (2) guarantees that a request to content c is not routed to a target RCS unless it is replicated beforehand. Constraint (3) ensures that a request is not routed to a MCS where the content does not exist. Constraint (4) indicates the fact that the total amount of contents replicated in a RCS must not exceed its storage capacity. Constraint (5) guarantees that a RCS can route the request for content c to at most one server, either a peer RCS or a MCS. The last constraint implies the binary nature of the optimization variables or constants.

In the following, we propose a relaxed C^{2}R^3 problem corresponding to the case where transmission delays are the dominant factors in a CDN network, i.e., $\tau_{ji} \gg \phi_j(Z_j)$, $\forall i,j \in RS$ and $\delta_{ji} \gg \phi_j(Z_j)$ $\forall i \in RS$ and $\forall j \in MS$. Thus, we assume $\phi_j(Z_j) \approx 0$ and the optimization problem is simplified to:

$$\text{Minimize} \sum_{c=1}^{K}\sum_{i=1}^{N} s_c d_{ci}\left(\sum_{j=1}^{N}\tau_{ij}x_{cij} + \sum_{j=1}^{M}\delta_{ij}y_{cij}\right) \tag{7}$$

The constraints (2)-(6) also apply to the optimization problem (7). Both problems (1) and (7) can be reduced to a capacitated facility location problem (CFLP) which is known to be NP-hard. The major difference between problems (1) and (7) and CFLP is that the former problems do not include facility opening cost. Nevertheless, this does not necessarily reduce the hardness of problems (1) and (7).

Suppose that the optimal C^{2}R^3 policy obtained by (7) is Z^* and the corresponding replication and request routing variables are $\{x^*_{ci}\}$, $\{x^*_{cij}\}$ and $\{y^*_{cij}\}$. Then, the average delay per chunk, represented by $\bar{\tau}$, perceived by a user is obtained by applying Z^* to (7) and normalizing, i.e.,

$$\bar{\tau} = \frac{\sum_{c=1}^{K}\sum_{i=1}^{N} s_c d_{ci}\left(\sum_{j=1}^{N}\tau_{ij}x^*_{cij} + \sum_{j=1}^{M}\delta_{ij}y^*_{cij}\right)}{\sum_{c=1}^{K}\sum_{i=1}^{N} s_c d_{ci}} \tag{8}$$

3.2.3 Distributed cooperative algorithms

For the problem defined by (7) and the set of constraints (2)-(6), we propose two distributed cooperative algorithms differing only by the content valuation method used. In the first algorithm, the local value of a content c in a RCS_i is defined as $s_c d_{ci}$, i.e., the product of size and the demand rate for c in RCS_i. In the second method, the value of c in RCS_i is equivalent to its popularity density defined as $\frac{d_{ci}}{s_c}$. We term these methods cooperative popularity-size product (CPSP) and cooperative popularity-density (CPD) schemes, respectively. Cooperation is realized by two major mechanisms: i) exchange of contents between peer RCSs, and ii) a system of aggregate content valuation to support the decisions on caching a requested content and evicting other cached items. Several data structures are implemented in each RCS to maintain information of different kinds exploited to facilitate the operation of the distributed algorithms. A content map (Ω), similar to a distributed hash table, is implemented in RCS and contains information about the location of content objects. When a content is replicated (or evicted) in a RCS, it informs the peer RCSs to update their maps. Two other maps are implemented to maintain information about the current incoming and outgoing request routing configuration. We term the former map RI and the latter RO. In the RO map, the RCS retains, for each non-cached content, the identity of the destination server chosen to serve the requests for that content. In the RI map, the RCS retains the identity of contents with exogenous demands, i.e., demands routed from other RCSs. Another map, denoted by T, maintains the transmission delays between each pair of servers. In the rest of this work, we assume the network delays are fixed. Nevertheless, it is not hard to extend the algorithm to operate under dynamic network conditions. This, for example, can be realized by using short probing signals periodically broadcast by individual RCSs, and sharing the new delay information with the peer RCSs.

When a RCS_i receives an end-user's request for a content c, initially, it updates the demand rate of c and performs a local cache lookup to reply the request. If it fails, the content map Ω is looked up to find a server (either a MCS or a peer RCS) having the content c in its cache and offering the shortest transmission delay. Let's denote such a server by R_c. Content c is retrieved from R_c which is then delivered to the user. At this point, RCS_i should decide whether or not to cache c. If there is sufficient cache space, the content is cached, otherwise the main process of the algorithm is triggered as follows. Recall that caching a new content c implies evicting some previously cached items, and thus deciding on caching c involves an estimation of the benefit in doing so. To this aim, RCS_i continuously estimates the aggregate values of the locally cashed items. This process melts down to the estimation of the aggregate demand rate per content, which is then applied together with the content size to obtain the aggregate value of the content. Denote the aggregate demand vector by $\Psi_i = \{\psi_h | h = 1,2,\dots H\}$, where H is the total number of contents cached in RCS_i, and ψ_h is the aggregate demand rate of content h obtained as follows:

$$\psi_h = d_{hi} + \sum_{j=1}^{\|P_h\|} \tilde{d}_{hj} \qquad (9)$$

where d_{hi} is the local demand rate of h in RCS_i, and P_h is the set of RCSs that route their local requests for the content h to RCS_i. A $RCS_j \in P_h$ is represented by an index $j \in \{1,2,\cdots \| P_h \|\}$, where $\| P_h \|$ is the cardinality of set P_h. Accordingly, $\tilde{d}_{hj}$ is an estimation of the actual demand rate d_{hj} of content h in RCS_j. The estimated rate $\tilde{d}_{hj}$ is obtained in RCS_i using the history of requests for content h routed by RCS_j to RCS_i. More specifically, RCS_i looks up its map RI to determine the set P_h of client RCSs. Then, for each member of P_h, the demand rate for content h is retrieved from the demand history retained for a sufficiently wide window of time. Once the vector Ψ_i is determined, RCS_i constructs an aggregate value vector $W_i = \{w_h | h = 1,2,\dots H\}$, where $w_h = \psi_h s_h$ and $w_h = \frac{\psi_h}{s_h}$ depending on whether CPSP or CPD method is used. It is worth mentioning that the demand estimation process is passive and imposes zero communication overhead.

We adopt a relatively different procedure for the estimation of the aggregate demand rate of a new content c subject to caching in RCS_i. Bearing in mind that no demand history is available for c in RCS_i, the estimation of the aggregate demand rate comes with some communication cost due to a simple probing mechanism employed in a procedure described below. Initially, we define the aggregate demand rate of c as $\psi_c = \sum_{j=1}^{\|Q_c\|} d_{cj}$, where d_{cj} is the demand rate for c in a $RCS_j \in Q_c$, and Q_c is the set of candidate RCSs satisfying three conditions: i) they have non-zero demand for c, ii) c does not exist in their local caches, and iii) RCS_i is the shortest server to such RCSs among those servers (including MCSs) having c in their local caches. To determine Q_c, RCS_i consults its content map Ω and splits the set of RCSs into two partitions Q_c^+ and Q_c^-. In the former set (Q_c^+), the member RCSs have previously

cached c, while, in the latter, c has not been cached. In the next step, RCS_i probes the members of Q_c^- to identify those with non-zero demand for c. Let Q_c^{-+} denote the set of such RCSs. Finally, RCS_i consults its delay map T to identify to which members of Q_c^{-+} it is the shortest RCS compared to the members of Q_c^+. Enforcing this last condition determines the set Q_c.
Given the aggregate demand and value vectors Ψ_i and W_i for the cached contents in RCS_i and the aggregate demand rate ψ_c of the new content c calculated as above, the eviction process reduces to solving the following ILP

$$\text{Minimize} \quad \sum_{h=1}^{\|H_i\|} W_i(h)x_h \tag{10}$$

$$\text{Subject to:} \quad \sum_{h=1}^{\|H_i\|} s_h x_h \geq s_c \;\&\; x_h \in \{0,1\} \tag{11}$$

where H_i is the set of cached items in RCS_i, s_h and s_c are the sizes of items h and c, respectively. x_h is a binary decision variable. We propose a greedy heuristic mechanism described in Figure 44 to approximate (10).

Data: Ψ: aggregate demand vector, W: aggregate value vector,
S: size vector. ψ_c, s_c: aggregate demand and size of the new content c.

Result: set E of *"to be evicted"* content items
$W_a \leftarrow sortAscendingOrder(W)$
$\Psi_a \leftarrow rearrange(\Psi, W_a)$, $S_a \leftarrow rearrange(S, W_a)$
$E \leftarrow \emptyset, h \leftarrow 1$
$sumDemand \leftarrow \Psi_a(h), sumSize \leftarrow S_a(h)$
while $sumSize \leq s_c$ & $h \leq \| W_a \|$
    $E \leftarrow add(h)$
    $h \leftarrow h + 1$
    $sumDemand \leftarrow sumDemand + \Psi_a(h)$
    $sumSize \leftarrow sumSize + S_a(h)$
end
if $sumSize \geq s_c$ & $sumDemand \leq \psi_c$ **then**
    $evict(E)$ & $cache(c)$
else
discard content c

Figure 44: Greedy content eviction algorithm

In the algorithm, we dropped the RCS index for the sake of readability. The RCS, initially sorts the cached items in ascending order of aggregate values and generates a new vector W_a. Then, the aggregate demand and size vectors are rearranged to conform to the sorted vector W_a. The "while" loop iterates through W_a and updates the accumulated size and demand rates until the point where the new content c can be accommodated.

3.2.4 Simulation results

The CDN topology used for evaluation of the proposed algorithms consists of 5 RCSs and 3 MCSs. A pool of 1800 content items with sizes uniformly distributed in a range of 1-52 chunks is considered. The content pool is equally distributed among the MCSs. Each content item is only available in one of the MCSs. Five RCSs with cache size of $\beta.ContentPoolSize$ chunks are considered where β is the ratio of RCS cache size to the total size of content pool. The range of β is $[0.01,0.11]$. The delay corresponding to the transmission of a chunk between RCSs and between MCS-RCS pairs are configured as shown in Table 12 and Table 13, respectively. Requests arrive at the system with a Poisson distribution with a mean arrival rate of 1000 requets per unit of time. The requests are assigned non-uniformly to RCSs, where the probability of assignment is defined by a parameter θ configured for individual RCSs as shown in Table 14. The popularity of contents in RCSs are considered to be Zipfian distributed with skew exponent α. For the performed simulations $\alpha \in [0.5,1.7]$. The simulation time is set to 1000 seconds. The results shown hereafter correspond to CRRR policies at the time point when simulation ends. The optimal solutions are obtained by solving problem (7) using CPLEX/Gurobi.

We compare the proposed popularity-based methods (i.e., CPSP and CPD) with a cooperative least-recently used scheme (CLRU), which is an LRU-based method extended to support cooperation. CLRU is similar to CPSP and CPD with an exception that, in the former, the local value of a content in a RCS is defined as the latest time the content was requested in the RCS. Accordingly, the aggregate value of a content is the latest time it was requested in the entire system. We also compare CPSP with CPD to find out which one is the better choice among the popularity-based methods.
The average delay per chunk is chosen as the evaluation metric (8). Evaluations are performed by fixing the Zipfian skew exponent and varying cache size parameter β and vice versa. The detailed descriptions of scenarios are shown in Table 15. The performance of algorithms are compared with the benchmark optimal solutions corresponding to the various scenarios.

Table 12: Inter-RCS delay matrix

	RCS1	RCS2	RCS3	RCS4	RCS5
RCS1	0	1	2	4	2
RCS2	1	0	3	5	3
RCS3	2	3	0	1	5
RCS4	4	5	1	0	2
RCS5	2	3	5	2	0

Table 13: MCS to RCS delay matrix

	RCS1	RCS2	RCS3	RCS4	RCS5
MCS1	7	9	10	7	8
MCS2	8	7	11	7	9
MCS3	9	8	9	11	11

Table 14: Demand distribution of RCSs

	RCS1	RCS2	RCS3	RCS4	RCS5
θ	0.3	0.15	0.15	0.1	0.3

Table 15: Scenarios w.r.t parameter ranges

Scenario	α	β
S1	0.5	[0.01,0.11]
S2	1.7	[0.01,0.11]
S3	[0.5,1.7]	0.01
S4	[0.5,1.7]	0.1

In Figure 45, the performance of algorithms are compared with respect to popularity distribution. Figure 45 (a) depicts the case where $\alpha = 0.5$, corresponding to a long tail popularity distribution. Thus, apart from a very small number of contents, the demand rates of various contents are highly similar and non-negligible. As shown in the figure, in the entire range, CPSP outperforms CLRU and CPD. In the case that the cache sizes are much smaller than the active content pool size ($\beta \leq 0.02$), CPSP (and CPD) are very close to optimal. CPD outperforms CLRU only for small cache sizes ($\beta \leq 0.05$) and loses its lead when the cache size grows. To explain the reason why the popularity-based methods outperform CLRU, especially for small cache sizes, recall that CLRU always replicates the most recently requested content and evicts as many least recently requested contents as necessary to accommodate the new content. Thus, it does not consider the overall popularity of contents over time. As a result, when the cache size is small, a content that is highly popular over the course of simulation time can be evicted from a RCS only because it was not requested recently. As another observation from Figure 45 (a), CPSP yields better performance than CPD. This is explained by the difference between the content valuation methods used in the two algorithms. In CPSP, as soon as sufficient space for large contents with high popularity becomes available, it yields a significant gain compared to CPD.

Figure 45 (b) illustrates the scenario where the Zipfian exponent is increased to $\alpha = 1.7$, corresponding to a short tail Zipfian distribution. In this scenario, the popularity-based methods considerably outperform CLRU when cache sizes are relatively small compared to the active content pool size. An explanation for this behaviour is that, with large α values, the content population

becomes highly bipolar. Thus, the popularity based methods are more likely to accurately distinguish between popular and non-popular items. However, CLRU is likely to evict very popular contents from a RCS to accommodate very unlikely ones because it relies on the request patterns in a recent time period, which may not be long enough to capture the actual popularity of contents. As the cache sizes grow, the cooperation among RCSs leads to a significant reduction in the performance gap between the popularity-based and CLRU algorithms. Furthermore, observe that the performance of CPSP and CPD are very similar due for the reason mentioned above regarding a bipolar popularity regime.

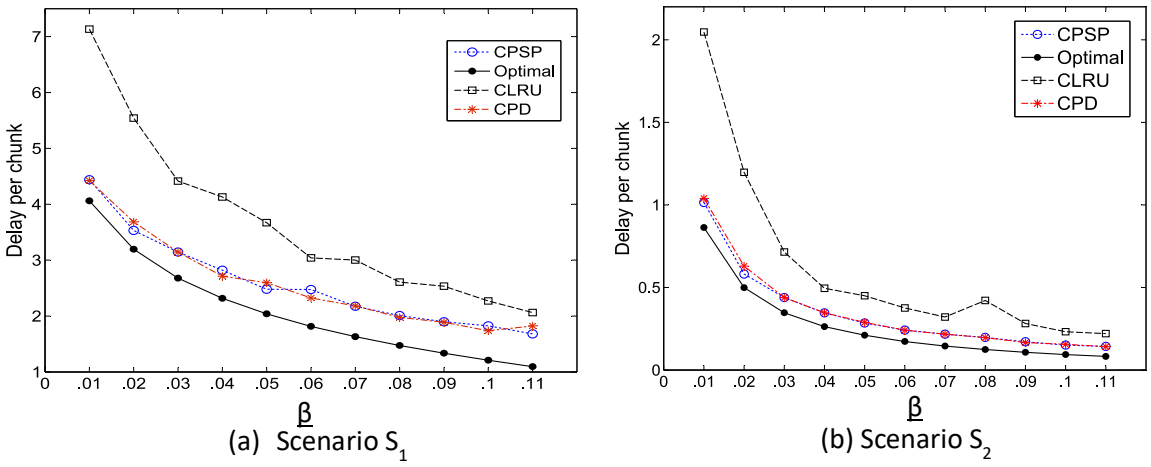


Figure 45: Impact of popularity distribution

In the following, we compare the performance of algorithms with respect to the cache size parameter and demonstrate the results in Figure 46. For a very small cache size corresponding to $\beta = 0.01$, as shown in Figure 46 (a), popularity-based methods exhibit very similar performances, but they significantly outperform CLRU in the entire range of α values. Also, the popularity-based methods demonstrate highly stable behaviour in the entire range. By contrast, CLRU is noticeably unstable and highly sensitive to the variation of popularity distribution. We increase β to 0.1 — ten times larger than the previous scenario — and demonstrate the results in Figure 46 (b). In this scenario, CPSD outperforms other methods, and CPD maintains a better gain than CLRU except for very small α values (i.e., $\alpha < 0.6$). Also, observe that the performance difference between popularity-based methods on one side and the recency-based method on the other side is markedly smaller than the previous scenario (i.e. S3). Further insights can be obtained by comparing scenarios S4 (Figure 46 (b)) and S2 (Figure 45 (b)). Although the average performance of algorithms is worse in S4 compared to S2, the convergence behaviour shows an opposite trend. From Figure 45 (b) we observe that, given a fixed large α, increasing the cache size has a limited impact on convergence rate of algorithms to the optimal solution. By contrast, given a large cache size, as shown in Figure 46 (b), the algorithms exhibit a sharp convergence to the optimal solution when the tail of popularity distribution shrinks, i.e., when α increases.

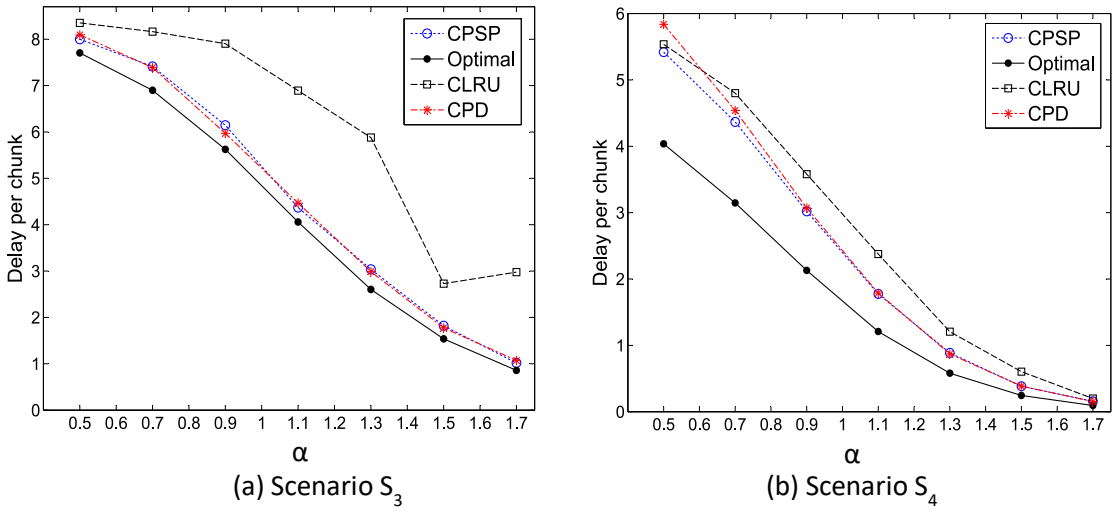


Figure 46: Impact of cache size

To obtain a holistic insight into the features of the algorithms, we present error percentages and delay quantities in Table 16 and Table 17, respectively. Errors are the percentage differences between delay values obtained in the simulation and those obtained by the optimal solution in each scenario. According to error results, CPSP and CPD exhibit their best performance in scenarios S1 and S3. Bearing in mind that these scenarios correspond to long tail popularity and small cache sizes, arguably they are the most likely real world situation. By contrast, the algorithms demonstrate the worst accuracy in scenarios S2 and S4, corresponding to short tail distribution and large cache size, however, Table 17 shows that the experienced delay is very small in these scenarios. Examining scenario S1 again reveals that although all algorithms demonstrate remarkable accuracy, the access delay is the largest among all scenarios. This observation has a general implication that the optimization of software aspects may not be sufficient if content providers wish to provide users with quality experience. Thus, a complementary approach is to increase the infrastructure capacity. Arguably, this scenario is highly likely in reality, considering the fact that a huge number of user-generated contents with non-negligible popularity are published in the Internet [59]. As final remarks, the best existing approximation algorithm to FLP with complete information is 1.52 optimal, and the worst-case result, found by simulations, for the proposed CPSP indicates 1.68 optimality. Furthermore, we propose CPSP as the preferred C^2R^3 mechanism since it outperforms CPD and CLRU in all scenarios.

Table 16: Performance evaluation - error (%)

Scenario	CPSP		CLRU		CPD	
	mean	stdev	mean	stdev	mean	stdev
S1	18	11	24	10	27	12
S2	41	19	134	42	42	18
S3	12	6	84	82	13	7
S4	48	8	88	40	50	8

Table 17: Performance evaluation - raw delays

Scenario	CPSP		CLRU		CPD	
	mean	stdev	mean	stdev	mean	stdev
S1	6.4	0.9	6.7	1	6.75	0.8
S2	0.5	0.4	0.8	0.8	0.5	0.5
S3	4.5	2.7	6.1	2.4	4.6	2.7
S4	2.3	2	2.6	2.2	2.4	2.2

3.2.5 Numerical study of power consumption

In this section, we show, by numerical study, that the proposed cooperative content replication and request routing solution also results in significant power saving in the transmission network of the overlay CDN. To simplify our analysis, we use an alternative interpretation of the delay metric which is

based on the number of hops. Therefore, the entries of Table 12 and Table 13 are regarded as the number of hopes between the pairs of RCSs and RCS-MCS, respectively. Each hop, in turn, is terminated by a pair of core routers as the constituent elements of the CDN's underlying transmission network. To employ such an interpretation, we consider an arbitrary core network that only requires realizing the hop distances given in the aforementioned delay matrixes in Table 12 and Table 13. Therefore, the exact topology of the network is not required to be known. Additionally, the delay values achieved by the three cooperative algorithms and the optimal solution (see Table 17) are also regarded as the average number of hops that should be traversed in the core transmission network to fetch a requested content. Note that, in our analysis, we do not take into account any hops traversed in the access and aggregation networks. This setting allows us to directly use the energy consumption model of conventional CDNs proposed in a seminal study conducted by Guan et. al. [10]. More specifically, we use the following expression proposed in [10] for calculating the energy consumed to deliver a content with a given rank k:

$$E_{tr} = 4B_k R_k \left[(p_d^r + p_d^{oxc}) \left(A \left(\frac{N}{n}\right)^{\alpha} + 1 \right) + p_d^{wdm} A \left(\frac{N}{n}\right)^{\alpha} \right] \quad (12)$$

where coefficient 4 represents cooling overhead, B_k is the size of content with popularity index k, R_k is the request rate of this content, p_d^r is the energy consumption in Joules per bit (J/bit) of a core router, p_d^{oxc} is the energy consumption (J/bit) of optical cross connect, p_d^{wdm} is the power consumption (J/bit) of WDM link, and the compound term $A\left(\frac{N}{n}\right)^{\alpha}$ is the optimal hop distance. The configuration of the energy consumption parameters, borrowed from [10], are as follows: $p_d^r = 1.2 \times 10^{-8}$, $p_d^{oxc} = 1.95 \times 10^{-8}$, and $p_d^{oxc} = 1.48 \times 10^{-9}$. Equation (12) is simplified further by eliminating the last term of Equation (7) in [10], corresponding to the energy consumption of the access network elements. The term $A\left(\frac{N}{n}\right)^{\alpha}$ in Equation (12) is replaced by the entries of Table 17, corresponding to the optimal hop distance achieved by the proposed algorithms and the optimal solution. We assume there are 10000 requests per hour for 1800 contents with a similar size of 20MB. The cache sizes and popularity distribution are configured according to the four scenarios described in Table 15, with a difference that we use the exponent $\alpha = 1.1$ for scenarios 3 and 4 (this is the mean exponent in the range $[0.5,1.7]$). Finally, to conform to the model described by Equation (12), we select a content with a certain popularity index, and conduct the analysis for this content. To this aim, we use the most popular content in each scenario. Recall that the hop distances to fetch this content are assigned with the entries of Table 17. In this sense, the power consumption measure corresponds to the worst case analysis (upper bound) because the hop distance for the most popular content is in effect lower than the mean values given in Table 17. This approach will inherently allow the consideration of cache size effect in the presence of the most popular content together with other contents.

Figure 47 illustrates the power consumption of the three cooperative algorithms and the optimal solution. Observe that the minimum power consumption is achieved in the first scenario, representing a heavy tail popularity distribution (corresponding to user generated contents). The second best power consumption behaviour is observed in scenario 4, where the cache size is large enough to accommodate larger portions of popular contents. When the cache size is small (in scenario 3) or the popularity distribution is short tail (in scenario 2) the power consumption becomes larger. Overall, this analysis indicates that the small cache size has the worst effect while the popularity distribution, if small, demonstrates the best effect on power consumption.

The power efficiencies of the proposed algorithms, expressed as the ratio of the optimal power consumption and the power consumption of the proposed algorithms, are illustrated in Figure 48. Observe that CPSP exhibits the highest efficiency in all scenarios, CPD is nearly as efficient as CPSP, while the CLRU is the least power efficient algorithm. Recall that the same behaviour was observed in delay efficiency as discussed in the previous section. We conclude that CPSP is the most efficient algorithm in terms of both delay improvement and power saving.

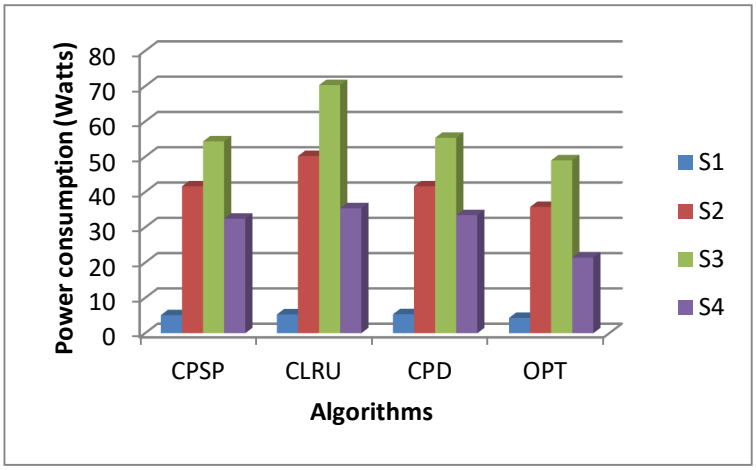


Figure 47: Power consumption of the most popular content in the proposed algorithms

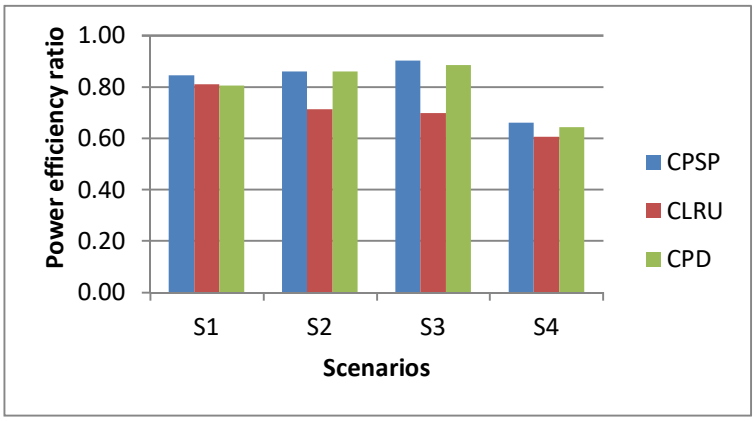


Figure 48: Power efficiency index

3.2.6 Conclusion and future work

We developed an optimization problem for minimum delay C^2R^3 problem, and popularity-based distributed algorithms are proposed to approach optimal replication and request routing policies. Our results, quantified by a delay-relevant metric, reveal that the proposed methods perform close to optimal in the most critical real-world scenarios characterized by a small cache size and long tail popularity distribution. In non-critical scenarios, the algorithms demonstrate a reasonable accuracy, small experienced delay, and outperform CLRU in almost all scenarios. It is also highlighted that in scenarios with a long tail popularity distribution, optimizing the software aspect of CDNs may not be sufficient for a quality user experience, and thus infrastructure capacity must be enhanced, too. We also showed, by numerical study, that our proposed cooperative algorithms achieve outstanding power efficiency ratios ranging from 60% to beyond 80%. Our future work includes the extension of the model to account for processing delay in the servers and also an extension to proactive replication policies.

3.3 Energy Efficient Placement of User Generated Content (UGC)

3.3.1 Introduction

As the long-prevailing model of a few dominant media production companies has changed to include an increasing fraction of user generated content (UGC), the Internet infrastructure is also impacted. The increase in UGC is creating a new usage pattern in the Internet with a shift towards content generation, distribution and sharing. In 2015, YouTube and Facebook, making up the largest share of UGC, accounted for a 35% and 20%, share of Internet traffic respectively, for mobile and fixed traffic in North America [60]. The global trend also shows that annual IP traffic surpasses 88 exabytes per month in 2016 and the increase is projected to be nearly threefold during the period 2015–2020 [53]. The UGC traffic volume is therefore growing rapidly.

The current best practice to cope with the demand of Internet traffic is by applying commercial Content Delivery Networks (CDNs). The lack of control from network operators over this over-the-top (OTT) traffic, apart from economic conflict [61], has caused increased congestion levels on their networks which can lead to degraded quality of experience (QoE) of their subscribers. Despite these strains, network operators are marginalized in the revenue chain of content delivery which is currently monopolized by few commercial CDN owners. This has motivated network operators and ISPs to look for ways of deploying and managing their own content delivery infrastructures, referred to as Telco CDNs [62] [63]. In addition to the monetary advantages from being involved in the revenue chain and an enhanced control over the traffic matrix, ISP-managed content delivery is inherently suitable for UGC due to the following: first, the trajectory of UGC is bottom-up, meaning that UGC is generated and uploaded by subscribers in the network operator domain. Second, UGC exhibit strong locality attributes, evidenced by recent studies on social networks showing that approximately 84% of the social community of an average Facebook user is collocated in the user's country of residence [64]. This implies that a substantial fraction of the total demand for UGC is likely to come from subscribers within the footprint of the network operator and from its local neighbors. By serving this demand locally, the network operator reduces its costs by enforcing appropriate traffic engineering and content management policies.

With ISPs involved in the content delivery, the economy of scale implies that, for any content uploaded by a local subscriber, an ISP must serve the external demand originated from users of other ISPs in addition to the local demand. Since a high volume of local UGC with potentially high demand can emerge, the resource capacity of the ISP may not be sufficient to accommodate and serve all content. Another main concern is the increased system complexity when a set of multiple ISPs with mutual demands are involved in content delivery. The complexity arises from the need for coordination among the participating ISPs. Furthermore, UGC, in contrast to most commercial content, is generated and uploaded by end users who do not have any commitment for retaining a local copy of the content which otherwise could be retrieved for replication in commodity content servers. Consequently, efficient resource assignment to UGC calls for online techniques. Despite these challenges, we note that for ISPs, UGC offers an important property in that their demands can be estimated from the social context of the users who upload and share the content, without additional exchange of information and coordination among ISPs. In this study, we exploit this property and address UGC delivery in a setting with multiple ISPs. We derive model with the objective to minimize power usage. The theory comprises a comprehensive formulation of the studied problem and an online algorithm which enables each ISP to individually decide which content should be placed and served locally. This algorithm takes advantage of the aforementioned property of UGC to determine the content demand, and also reduces the system complexity by eliminating inter-ISP coordination and information exchange. We also note that while this work focuses on power usage as the objective of optimal UGC placement, the approach taken and the accompanying principles can be applied to other relevant cost functions such as congestion and delay.

3.3.2 System modelling

We consider a network topology according to Figure 49 with several ISPs belonging to a set $I = \{1,2,..., I\}$. It is assumed that each ISP maintains its own storage in order to keep user generated content locally. If a user uploads his/her content, then in the simplest case, there are two scenarios; either to upload the content to a social network provider's central server or to upload content locally to the ISP's storage (with limited storage space and other resources).

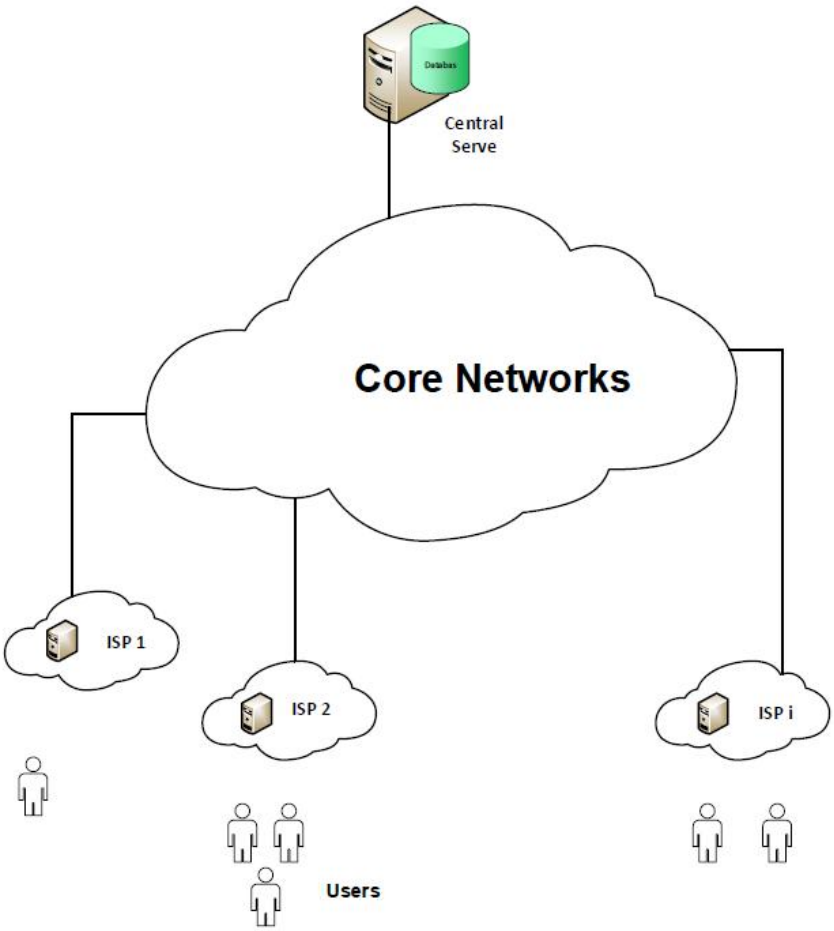


Figure 49: The network topology

We model the system from a content distribution perspective. There is a set of objects $\mathrm{N} = \{1,2,..., N\}$ where each object instance is denoted by $n \in \mathrm{N}$. The popularity of each object can be derived from the user's social relationship and in this work, the request rate (the number of requests per second) is a function of the number of users' friends since they are usually the first users who request the content. The decision can be made at the end of a certain amount of time periodically which lets us focus on a single decision problem. For simplicity, we assume that all objects are of the same size. The total request rate of object n is $\Lambda(n)$ and the requests come from different ISPs. The request rate of object n from ISP $i \in$ is $\lambda(n,i)$ and clearly $\Lambda(n) = \sum_{i=1}^{I} \lambda(n,i)$. There is a set $\mathrm{N}(i) \subseteq \mathrm{N}$ which denotes the subset of all contents with the source being ISP i (and $\bigcup_{i \in I} \mathrm{N}(i) = \mathrm{N}$ and we also assume the intersection of all $\mathrm{N}(i)$ is empty set. Then, for each content, there are two types of requests, local requests and remote requests. The local request rate is defined as $\lambda(n,i)$ s.t. $n \in \mathrm{N}(i)$ and the remaining remote requests are from other ISPs which equal to $\sum_{j \in I} \lambda(n,j)$ s.t. $n \in \mathrm{N}(i)$ and $j \neq i$ and these conditions denote the total requests of all ISPs other than i for object n which belongs to ISP i. This remote request rate is denoted by $\mu(n,i)$ s.t. $n \in \mathrm{N}(i)$.

3.3.3 Power usage minimization problem

The corresponding decision problem is modeled using binary integer programming. The main optimization variable is x_{ni}, defined as:

$$x_{ni} = \begin{cases} 1 & \text{if object } n \in \mathsf{N}(i) \text{ is decided to store in } ISP_i \\ 0 & \text{o.w.} \end{cases}$$

Each ISP can decide to place object n in its server locally, or place it on the central server where the replication of content is mutually exclusive i.e. cannot take place on both servers. The reason for this assumption is that UGCs usually is short-lived and replication can lead to increased complexity and resource consumption. The second assumption is that ISP i does not have any control over other ISPs and it cannot place its own objects on other ISPs' servers. It is important to note that if $x_{ni} = 0$ then it doesn't necessary mean object n is stored on a central server. Assume the source of object n is ISP j then, for all ISPs other than j, these decision variables are equal to zero. So, if $x_{ni} = 0 \text{and} n \in \mathsf{N}(i)$ then it can be concluded that object n is stored on a central server. This can also be expressed as follows: object n is stored on a central server if $\sum_{i \in I} x_{ni} = 0$.

We define power usage as an optimization cost function. Each downloaded object is transmitted over some links and over network nodes that use energy. In the transmission part, the usage is linearly dependent on the number of hops. Building on previous work in [32], the definition of the transmission energy usage per object is:

$$\xi(H) = 4B(e^r + e^{oxc})(H+1) + e^{wdm}H + 3e^e + e^g + 2e^{pe}) \quad (1)$$

where B denotes size of object in bits, H is number of hops and e^r, e^{oxc}, e^{wdm}, e^e, e^g and e^{pe} denote energy density (joule/bit) of a core router, optical cross connect, WDM link, Ethernet switch, gateway router and provider edge router. The factor 4 accounts for redundancy, cooling and other overheads and the factors 3 and 2 for Ethernet switches and edge routers stem from the internal topology of edge networks. For simplicity, we consider the same networking node types among all ISPs. Thus, the definition of the power usage cost function is:

$$\begin{aligned} Minimize \quad & \sum_{i \in I} \sum_{n \in N(i)} (\xi(H(i))\lambda(n,i)x_{ni} \\ & + (\xi(H^s(i)) + \xi(H(i)))\lambda(n,i)(1 - x_{ni}) \\ & + \sum_{j \in I \setminus i} \sum_{m \in N(j)} (\xi(H(i,j))\lambda(m,i)x_{mj} \\ & + (\xi(H^s(j)) + \xi(H(j)))\lambda(m,i)(1 - x_{mj})) \end{aligned} \quad (2)$$

$$Subject\ to \ \text{constraints}\ (3)(4)(5)(6)(7)$$

where $H(i)$ is average hop distance between internal server of ISP i and its users, $H(i,j)$ is hop distance between servers of ISP i and j, finally, the hop distance between server of ISP i and central server denotes by $H^s(i)$. There are constraints on the way ISPs can make decisions. The storage size $S(i)$, available capacity of the transit link (the ISP link to the outside network) $C(i)$, and maximum rate of streams that can be served by the local ISP server $R(i)$ (the unit of C and R is content rate). The following constraints are applied to the optimization problem.

$$\text{Capacity constraint:} \quad \Big(\sum_{n\in N(i)} (\mu(n,i)x_{ni} + \lambda(n,i)(1-x_{ni})) + \sum_{m\in N\setminus N(i)} \lambda(m,i)\Big) \le C(i), \quad \forall i \in I \tag{3}$$

$$\text{Storage constraint:} \sum_{n\in N(i)} x_{ni} \le S(i), \quad \forall i \in I \tag{4}$$

$$\text{Stream constraint:} \sum_{n\in N(i)} \Lambda(n)x_{ni} \le R(i), \forall i \in I \tag{5}$$

$$\text{Local constraint:} \sum_{m\in N\setminus N(i)} x_{mi} = 0, \quad \forall i \in I \tag{6}$$

$$\text{Binary constraint:} \; x_{ni} \in \{0,1\}, \; \forall n \in N, \forall i \in I \tag{7}$$

As mentioned above, $C(i), S(i), \text{and } R(i)$ are constants and C and S measures the numbers of objects per second. The transit link has capacity $C(i) \ge (max(\sum_{n\in N(i)}(\lambda(n,i)(1-x_{ni}) + \mu(n,i)x_{ni}) + \sum_{m\in N\setminus N(i)}\lambda(m,i))$. Therefore, if an ISP decides not to place any content locally it should have sufficient link capacity to fetch content from central servers and conversely, if content is placed locally, then there should be sufficient link capacity to serve requests from other ISPs.

The above integer program with corresponding constraints places all N objects as well as minimize the total power usage for all ISPs globally.

3.3.4 Online Algorithm

Our goal is to place uploaded objects optimally and for this there are two possibilities. One way is by solving the corresponding optimization problem offline at the end of each period using algorithms such as branch and bound, followed by applying the simplex algorithm. However, this strategy is impractical for two reasons. First, usually, there is no central authority to force all ISPs where to place objects in order to reach optimality. Second, once a user wants to upload or share their content, this should be made available instantly in the corresponding online social networks. In order to meet these two challenges, it is necessary to define an online algorithm which can place each object one by one as upload requests happen within each ISP.

Online placement algorithms can work well in interactive systems. In practice, some applications such as online social networks, which have interactions between users, need to decide instantly for each upload (or sharing) request. Here, online algorithms are suitable because if the ISP already has information about its storage,capacity and stream rate for each period, then it does not need knowledge about future arrival uploading requests (their popularity etc.). Even so, if there is an accurate request rate predictor, there is still uncertainty about the order of upcoming contents, because the online decider does not know if the next object will be preferred over the object currently under decision or not (here "preferred" means if the object has higher popularity). If there is a restriction so that previous decisions on incoming objects cannot be changed, then it is impossible in many cases to reach the optimal solution. However, it is well known that competitive online algorithms can yield solutions near the optimum and the distance to the offline optimum can be bounded [65].

We propose an online algorithm that ISPs can run. In order to simplify our reasoning for clarity, we consider one ISP –which we refer to it as *target ISP*– with set of content Γ and localize (i.e. for one ISP) the integer program with a power usage cost function. The program is expressed as follows:

$$
\begin{aligned}
&Minimize \quad cost(x_n) \\
&s.t. \quad \sum_{n\in\Gamma}(\mu(n)x_n + \lambda(n)(1-x_n)) + \sum_{m\notin\Gamma}\lambda(m) \le C \\
&\qquad \sum_{n\in\Gamma} x_n \le S \\
&\qquad \sum_{n\in\Gamma}\Lambda(n)x_n \le R \\
&\qquad x_n \in \{0,1\}
\end{aligned}
\tag{8}
$$

This problem is a variant of a multiple knapsack problem but with some differences which renders it more difficult to solve. Online knapsack problems have been extensively studied in the literature [66] [67] [68]. These problems are usually maximization problems and in order to make our problem fit a knapsack problem better, the objective function (which is cost minimization) is translated to cost saving maximization. To that end, we formulate the power saving maximization objective function as follows: $\text{maximize}\sum_{n\in\Gamma}\xi(H^s)\lambda(n)x_n$. This objective function states that if an ISP decides to place an object n with local request rate $\lambda(n)$ locally, then it can save $\xi(H^s)\lambda(n)$ power from a global point of view, because if object n is placed on a central server, then this ISP should download it $\lambda(n)$ times per second and each time the retrieval of the object from the central server incurs an additional energy cost of $\xi(H^s)$.

In knapsack-like maximization programs, the value to weight ratio $v^w(n)$ of each object n plays an important role and is used as a foundation for many successful algorithms. Generally, this ratio represents the objective function increase (cost saving) for each object an ISP decides to place in its storage and from there serve requests. In our placement problem, the value of object n is $v(n)=\xi(H^s)\lambda(n)$ as mentioned above. It remains to calculate a unified weight for this object. To this end, we use the structure of the constraints in the optimization problem described by (8). Let $w^S(n)$, $w^R(n)$, and $w^C(n)$ represent the normalized weight elements corresponding to storage, server, and link capacity utilized by object n, if the ISP decides to place this object locally. Applying the storage constraint in (8) to object n only, we obtain $w^S(n)=1/S$. This yields for the server constraint $w^R(n)=\Lambda(n)/R$. For link capacity, the situation is non-trivial because in both cases of placing the object locally or remotely in the central server, there exists a weight (i.e. cost) on link capacity which, by exploiting the corresponding constraint in (8), becomes $w^C(n)=\frac{\mu(n)-\lambda(n)}{C-\lambda(n)}$. The unified weight of object n, also interpreted as its weight on the entire system, is thus defined as $w(n)=\max(w^C(n),w^S(n),w^R(n))$ which lies in the range $(0..1]$. Note that this range is valid so long as the weight of each object is less than or equal to the storage size, which holds true in the real scenarios. Although $w^C(n)$ may yield a negative value, the maximum of the three weights is always positive, and, accordingly, the value to weight ratio $v^w(n)$ is positive $\forall\, n$.

The value to weight ratio described above is used in our proposed online algorithm, demonstrated in Algorithm 1. The algorithm operation relies on the resource occupancy state, defined as the fraction of

a resource being occupied when a new placement decision is made. As can be seen from the algorithm, there are three state variables $C^{'}, S^{'}$ and $R^{'}$ (initialized in line 1 of the algorithm) and used to keep track of the occupancy of the link capacity, storage and server, respectively. The occupancy variables are then exploited to calculate two unified occupancy parameters f and $f(n)$ (lines 7 and 8 in Algorithm 1), with the former corresponding to the maximum occupancy among the three resources right before placing the object n while the latter is the maximum occupancy if the ISP decides to place the object n locally. The decision whether to place the object locally requires two conditions to be fulfilled, simultaneously (line 10); first, the local placement must be feasible (i.e. $f(n) \leq 1$). Second, the value to weight ratio of object n must be larger than a certain threshold, i.e. $v^{w}(n) \geq \Psi(f)$ where $\Psi(.)$ is a threshold function which assumes the current f as a parameter and determines a threshold. Decisions to store objects locally becomes more and more strict as the threshold increases. We adopt a threshold function proposed in [68] and defined as $\Psi(f)(Ue/L)^{f}(L/e)$ where U and L are upper and lower bounds of the value to weight ratio (v^{w}) and inferred by an ISP using information about content request rates already acquired, for instance, from the social context of users who upload the contents. Without loss of generality, we assume U and L are given in our algorithm. Once the placement decision is made, the placement and state parameters are updated accordingly (lines 11-14 for the case of local placement and lines 16-17 for remote placement). We note that between two consecutive placement decisions in a given ISP, chances are that other ISPs receive new contents (from their local subscribers) for which they make placement decisions. To account for the link occupancy caused by requests originated from our target ISP to these external contents, we introduce a variable $\lambda^{o} = \sum_{m \in \Gamma^{'}} \lambda(m)$ where $\Gamma^{'} \subset \mathsf{N} \setminus \Gamma$ is the set of contents placed by other ISPs since the occurance of the last placement decision in our target ISP. We use λ^{o} to update the occupied capacity of the link (see lines 12 and 17).

Algorithm 1 Online Content Placement Algorithm

Input: `content specific parameters:` local and external request rates (λ and μ) of each incoming content, upper bound (U) and lower bound (L) of the value to weight ratio.
`ISP resource parameters:` link capacity (C), storage capacity (S), server capacity (R)

Output: Binary placement decision $x_n, \forall n \in \mathcal{N}(i)$

1: $C' \leftarrow 0,\ S' \leftarrow 0,\ R' \leftarrow 0$
2: **for** each object n arriving **do**
3: $\quad \Lambda(n) \leftarrow \lambda(n) + \mu(n)$
4: $\quad v(n) \leftarrow \xi(H^{s})\lambda(n)$
5: $\quad w(n) \leftarrow$ **max** $\left(\frac{\mu(n)-\lambda(n)}{C-\lambda(n)}, 1/S, \Lambda(n)/R\right)$
6: $\quad v^{w}(n) \leftarrow v(n)/w(n)$
7: $\quad f(n) \leftarrow$ **max** $(\frac{C'+\mu(n)}{C}, \frac{S'+1}{S}, \frac{R'+\Lambda(n)}{R})$
8: $\quad f \leftarrow$ **max**$(C'/C, S'/S, R'/R)$
9: $\quad \Psi(f) \leftarrow (Ue/L)^{f}(L/e)$
10: $\quad$ **if** $f(n) \leq 1$ and $v^{w}(n) \geq \Psi(f)$ **then**
11: $\quad\quad x_n \leftarrow 1$
12: $\quad\quad C' \leftarrow C' + \lambda^{o} + \mu(n)$
13: $\quad\quad S' \leftarrow S' + 1$
14: $\quad\quad R' \leftarrow R' + \Lambda(n)$
15: $\quad$ **else**
16: $\quad\quad x_n \leftarrow 0$
17: $\quad\quad C' \leftarrow C' + \lambda^{o} + \lambda(n)$
18: $\quad$ **end if**
19: **end for**

It can be proven that this algorithm guarantees the distance to the optimum solution if $\forall n, L \le v^W(n) \le U$ as the algorithm is a $(Ln(U/L)+1)$-competitive online algorithm [68]. The larger the distance between L an U, the larger the distance to the optimum point. The knowledge of these bounds represents a practical challenge in itself. However, if the popularity distribution or friendship status information is available, then it is easy to predict these bounds from the previous period. From the results below, it can be seen how the online placement results are close to optimal.

3.3.5 Simulation Results

Simulation configuration

In this section, we show how the online algorithm performs compared to the offline global optimum and also as a benchmark, we include the case without any local placement. We wrote the offline integer program in AMPL and solved using CPLEX in Neos server [69]. The online algorithm was implemented in MATLAB.
We consider 5000 video objects with equal size of 1 MB where each object is uploaded by separate users. The network topology consists of 50 ISPs and each has its own server. There is also one central server (i.e. belonging to a content provider or OSN data center). The source of each object is selected randomly (uniformly) before the optimization run. It is assumed that the request demand follows a Zipfian distribution with exponent 1.05 for all objects with a maximum of ten requests per second for the highest ranking object (in accordance with today's case where some users have very large social networks). We define a set of scenarios and in each scenario, we consider a different probability for partitioning the total requests into total local and remote requests. For each objective function, there are 10 different local requests percentages (10% ... 100%). For the online algorithm, different orders of upcoming objects affect the decision performance. We therefore ran the algorithm 1000 times for different object orders. In order to make decisions, the ISP needs the lower and upper bound of the value to weight ratio and we assume that ISPs obtain this information from the previous period. In the simulation, we ran the algorithm for the whole period (i.e. all contents uploaded) to estimate the upper and lower bounds, then we ran the algorithm again when 5000 contents arrived one by one.
We also made assumptions for the constraints and the objective function. We assume symmetric numbers for each ISP and the available storage during one period was set to 50 objects while the server limit was set to ten content streams per second. The link capacity C was set to a large value (fifty content requests per second) to uphold the feasibility constraint. For the optimization parameters, we followed the values of energy density of devices from [4] and [9] and calculated transmission energy according to (1) for given hop distances. We considered $H(i) \in [1..4]$ following a discrete uniform distribution, $H(i,j) = H(i) + H(j) + 1$ and $H^s(i) \in [3..14$ also uniformly distributed. These values converge to their mean when repeating the simulation many times.

Results

In Figure 50 we show the ISPs' power usage in three different scenarios; the global optimal (offline integer program), online algorithm placement and a scenario without any local placement. The first and last scenarios show the best and worst cases in terms of power usage. We considered that on average, 30 percent of each ISP's requests are from local users. It should be noted that the energy usage for each object n requested by ISP i is considered as a part of ISP i's total power usage even if the object is located in the central server or at another ISP.

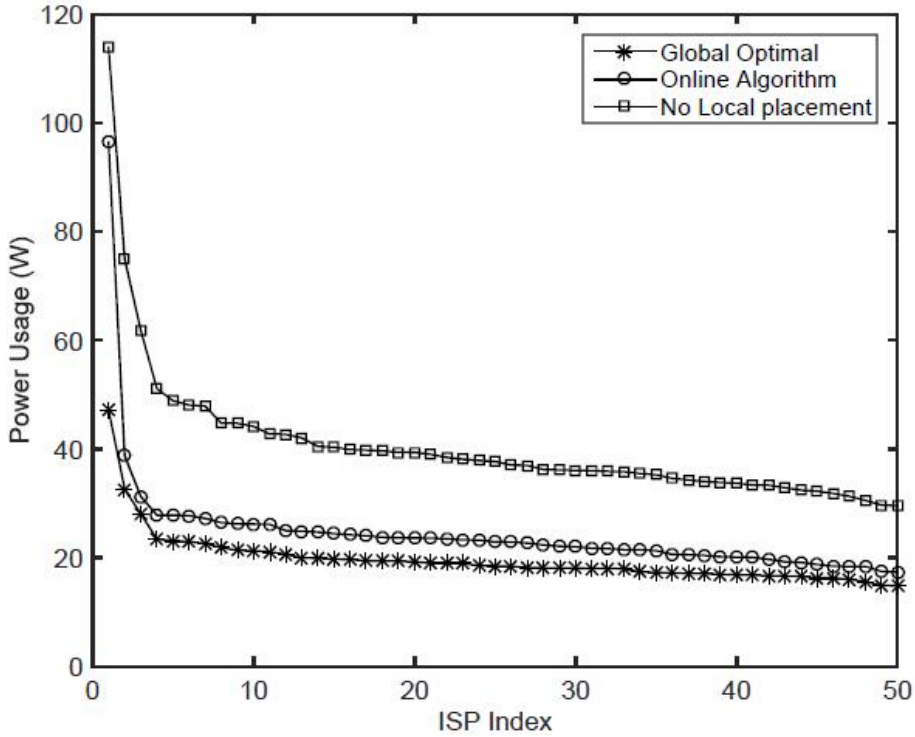


Figure 50: The power usage by each ISP with 30 percent local request

In this configuration, the achieved power usage is near the optimal global result. In the above figure we sort ISP's by the number of generated requests to clarify the result.

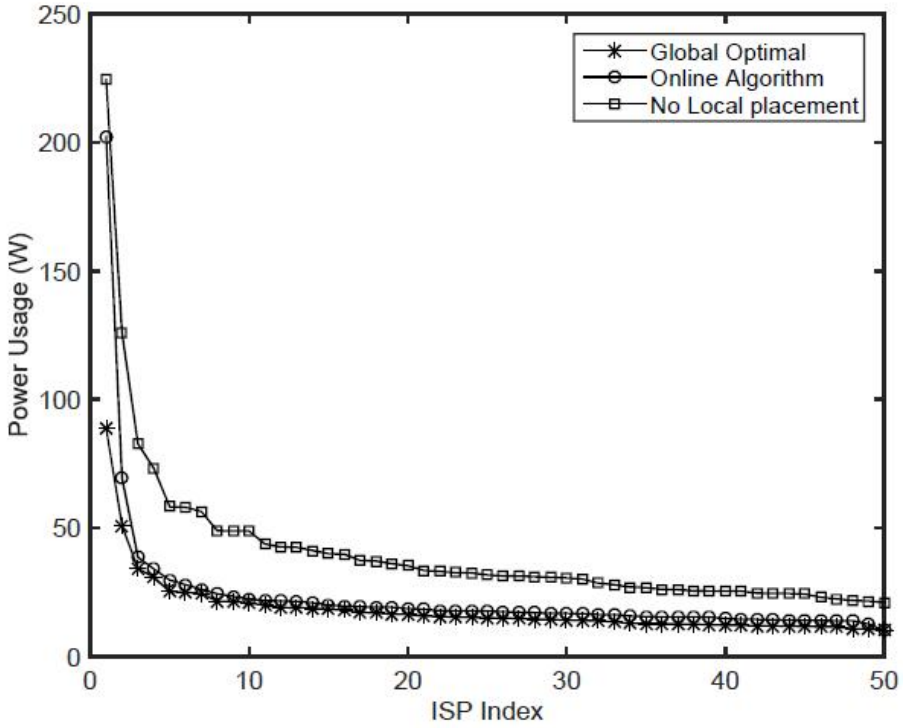


Figure 51: The power usage by each ISP with 70 percent local request

Figure 51 shows the same result with the assumption that the majority of requests are local (on average 70 percent). In this case, we can see even better results. Even when there is uncertainty about future uploaded objects, the algorithm performs well but for ISPs with a high portion of requests (ISP index 1, 2 and 3) the online algorithm does not perform as well. The reason is that these ISPs

have contents with extreme request rates and the online placement is very strict to keep them locally because of their weights.

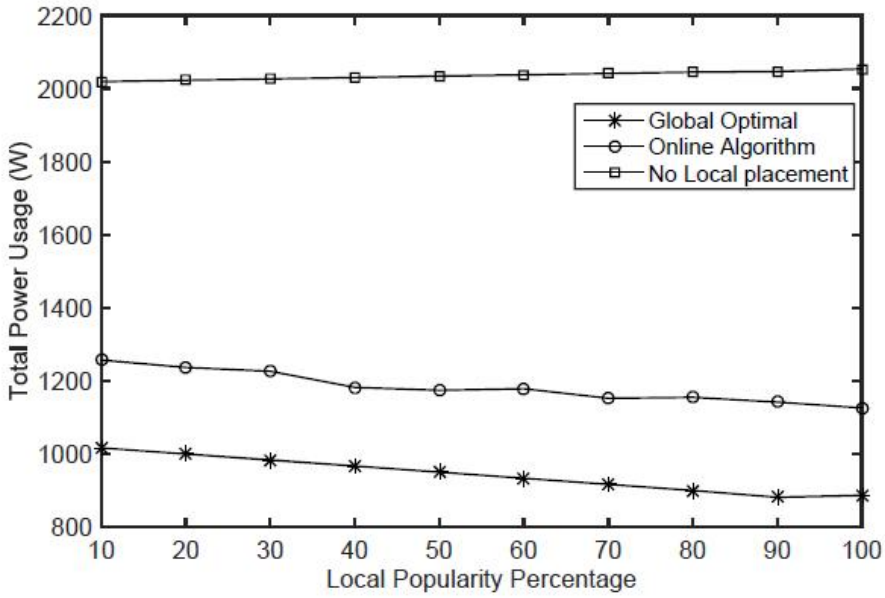


Figure 52: The total power usage by each ISP with different local request percentage

Figure 52 shows the total power usage of all ISPs in 10 different scenarios in which we have varying local popularity (10% ... 100%). The results show that the higher the local popularity is, the more power is used if there is no local placement but less power is used if optimal placement is performed. As mentioned above, the efficiency of the algorithm depends on the ordering of popularity among generated content. Figure 52 shows the average performance of 1000 different incoming orders.

3.3.6 Conclusion

While UGC is projected to make up a significant share of Internet traffic, we note that very little research attention has been dedicated to this type of content, particularly, when placed in the context of content delivery networks. This study is thus viewed as a first contribution to explore efficient solutions for the placement and delivery of UGC. To this aim, we develop a formulation of the UGC placement problem in a network setting comprising multiple ISPs. Also, as our main contribution in this study, we propose an online algorithm which relies on a valuable attribute of UGC, namely, its strong tie with social networking contexts, to determine the content demand and use it for efficient placement decisions. We advocate our choice of an online technique by recalling that UGC has uncertain persistence, and thus instant decisions must be made upon its exposure to the system. We showed that our proposed online solution performs close to an offline placement solution as a benchmark, while it imposes little or no overhead in terms of inter-ISP coordination and information exchange.

3.4 Trade-offs in energy usage and server stability in content delivery data centres

3.4.1 Introduction

Data Centers (DC) are an integral part of the Internet ecosystem. They house many types of services ranging from plain storage to more sophisticated computing and streaming services. DCs are increasing in number, thanks to the fast-pace evolution of modern services such as Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS). Content Delivery Networks (CDN) alone involve hundreds of thousands of data centers around the globe. Considering the tremendous capital expenditure for the deployment of such a large scale infrastructure, sustainable profitability entails saving in operational costs while maintaining a competitive service quality.

Data centers are notoriously high users of electrical power. This is attributed foremost to ventilation and cooling overhead, but the contribution of server equipment to the total power usage is also considerable. Specifically, in a data center, server equipment use significantly more energy than communication devices (servers about 40 % and networking devices about 10 %) [70]). Therefore, power saving in servers plays a prominent role towards the realization of green data centers. This problem space has attracted much attention in research lately and, among several approaches for power saving, dynamic server provisioning (DSP) has attracted the most attention of the investigated techniques. The operation of DSP is governed by its decisions about power on/off states of individual servers. Before deciding to power off a server, it is essential to assess the impact of this decision on service performance. Specifically, with the offered load fixed, when a server is powered-off, the remaining operating servers will face an increased load. This additional load gives rise to increased server response time due to additional queuing delay, and when the queuing delay grows unbounded (i.e. queue instability), this will result in service disruption.

In a data center with multiple clustered servers, load balancing is an existing solution to jointly save energy and enhance service performance – notably, service availability and response time– [6] [25]. Classical load balancing techniques offer a homogeneous response time among servers, but, when used for the purpose of DSP, their merit comes at the cost of a reduced degree of freedom in energy saving. Furthermore, an explicit load threshold is required to be set for servers in order to support DSP. To overcome these issues, in this work, we propose a load balancing technique based on Lyapunov stability analysis. We develop an optimization problem for DSP with the objective of energy usage minimization. This optimization framework is used together with a drift-plus-penalty technique to derive a load dispatching algorithm analytically. Our proposed algorithm, called Stable and Energy Optimal Load balancing (SEOL), aims at reaching a trade-off between server stability and energy usage. SEOL offers a high level of flexibility by allowing the desired trade-off behaviour to be described using a single parameter. Our comparison of SEOL with state-of-the-art load balancing techniques shows that SEOL has the potential to achieve a superior performance in terms of per-server experienced load and the total energy usage of all servers in the cluster.

The remainder of this section is organized as follows: Section 3.4.2 presents a stochastic optimization framework for minimizing energy usage with the constraint of server stability. Section 3.4.3 describes our proposed algorithm designed to reach a trade-off between energy usage and server stability. In Section 3.4.4, we compare our algorithm with several state-of-the-art load balancing techniques. Finally, we conclude in Section 3.4.5.

3.4.2 System modeling

We consider a model where there are K clusters and each cluster consists of N servers. Let N_i^k be server i at cluster k, $i \in N = \{1,2,3,\dots,N\}$ and $k \in K = \{1,2,3,\dots,K\}$. Each cluster has its own dispatcher (load balancer). Each dispatcher is responsible for disturbing the incoming requests among the servers in that cluster. In this work, we consider that each dispatcher receives a random number of arrivals and independently from other dispatchers makes a decision for the distribution of the requests. In the rest of the study, we therefore omit the cluster index k for notational simplicity.

We consider that load balancing is performed on discrete time slots $t = \{1,2,3,\dots,T\}$ with certain length τ seconds. Each server has a separate queue buffer to keep jobs (i.e. the number of requests assigned to server i) and the queue backlog of server i at time slot t is denoted by $q_i(t)$. Let $w(t)$

be the number of requests arrived at the cluster at time slot t and $x_i(t)$ fraction of $w(t)$ is assigned to server i where $0 \le x_i(t) \le 1 \ \ \forall i,t$. We denote $\alpha_i(t) = w(t)x_i(t)$ as the amount of the arrived requests at server i at t. We assume that server i has a fixed service capacity $\delta_i \ \ \forall i$. Then, the queue dynamic of server i is given as follows:

$$q_i(t+1) = [q_i(t) + \alpha_i(t) - \delta_i]^+, \tag{1}$$

where $[x]^+ \equiv \max[x,0]$. Energy usage in a cluster depends on the number of active servers processing the requests and the amount of load on each server (energy usage of cooling and other devices are not considered). We use a general model given in [16] for power usage for a server i is $p_i = p^{idle}(i) + (p^{peak}(i) - p^{idle}(i))l_i(t)$ where $l_i(t) \in [0,1]$ is the normalized load on server i, which is the ratio of the current load and the peak load of the server given as follows:

$$l_i(t) \equiv \frac{q_i(t) + \alpha_i(t)}{\delta_i}. \tag{2}$$

Since each slot lasts τ seconds then the energy usage of the server i is equal to $\xi_i(t) = p_i.\tau$. Then, the total energy usage in the cluster of N servers is given as $\xi(t) = \sum_{i=1}^{N} \xi_i(t)$. If server i is not overloaded, i.e., $0 \le l_i(t) < 1$ then the energy usage of server i in time slot t is equal to:

$$\xi_i(t) = \xi_i^{idle}\left(1_{\{\alpha_i(t)+q_i(t)\neq 0\}}\right) + \left(\xi_i^{peak} - \xi_i^{idle}\right)l_i(t) \tag{3}$$

where $1_{\{.\}}$ is the indicator function. Clearly, if $l_i(t) = 1$ then $\xi_i(t) = \xi_i^{peak}$. As a matter of fact, $\xi(t)$ is a function of $x_i(t), q_i(t)$ for all i and $w(t)$ but for notational simplicity we express it as a function of t.
Our aim is to find a dynamic algorithm that can distribute the incoming requests at each slot among the servers optimally so that the queue of every server is always bounded, i.e., $E[q_i(t)] < \infty$ and the minimum energy usage is attained. In other words, the dispatcher (e.g., load balancer) allocates the total requests $w(t)$ among all the servers with the goal of minimizing the total energy usage of servers and keeping the stability of all the queues in the cluster. The solution to the following stochastic problem provides the desired algorithm:

$$\textbf{Minimize: } \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[\xi(t)] \tag{4a}$$

$$\textbf{s.t : } \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[(w(t)x_i(t) - \delta_i)] \le 0 \quad \forall i \tag{4b}$$

$$\sum_{i=1}^{N} x_i(t) = 1 \quad \forall t \tag{4c}$$

$$0 \le x_i(t) \le 1 \quad \forall i,t \tag{4d}$$

The constraint in (4b) assures the stability of each server queue in the cluster and the second constraint in (4c) enforces that all the incoming requests at each slot are distributed among the servers without causing any blocking and request drop.
Before solving the above optimization problem, we want to point out some issues related to this problem. The objective function (4a) is nonlinear because of its piecewise definition. Specifically, there is a binary indicator function in the objective to decide if there is any request to serve or not, i.e., $1_{\{x_i(t)w(t)+q(t)\neq 0\}}$ which is not differentiable and makes it difficult to find a close-form solution. In order to overcome this problem, we approximate the indicator function by a linear function given as

$\left(\frac{x_i(t)w(t)+q_i(t)}{c_i w(t)+q_i(t)}\right)$ which is now differentiable and linear where $0<c_i\leq 1$ is a given constant. As long as the queue sizes are sufficiently high or both $x_i(t)$ and $q_i(t)$ are zero, the approximation is the same as the actual function. The main problem with this approximated function is that it may unnecessarily keep the server active which will result in higher energy usage, especially when the queue sizes are low. However, c_i plays a critical role in the decision of switching the servers off. Then, our new objective function is given by:

$$\xi_i(t) = \xi_i^{idle}\left(\frac{x_i(t)w(t)+q_i(t)}{c_i w(t)+q_i(t)}\right) + \left(\xi_i^{peak} - \xi_i^{idle}\right) l_i(t) \tag{5}$$

where $l_i(t)$ is given in (2). Next, we present our Stable Energy Optimal Load balancing (SEOL) algorithm which solves the problem defined in (4a)-(4d).

3.4.3 Energy Efficient Dynamic Algorithm

SEOL dynamically decides how much data should be allocated to each server at each time slot by considering energy computation and the queue size of each server. As input parameters, it takes the queue size information and server-specific information such as p_i^{idle} and p_i^{peak} and δ_i from each server and solves a convex optimization problem to determine the optimal $x_i^*(t)$. Details are given in Algorithm 1.

Algorithm 1: SEOL Algorithm

Input: Initially gets these values by control messages from servers $p_i^{idle}, p_i^{peak}, q_i(0), \delta_i, c_i, V$;

Output The optimum value of $x_i^*(t)$ for each i and t

for *at the dispacher for each time slot* t **do**

Observe $q_1(t), q_2(t), ..., q_N(t)$;

Observe $w(t)$ and solve the following sub-problem :

Minimize $V.\sum_{i=1}^{N}\xi_i(t) + \sum_{i=1}^{N}(\frac{1}{2}x_i^2(t)w^2(t) + q_i(t)x_i(t)w(t) - x_i(t)w(t)\delta_i)$

Subject to $\sum_{i=1}^{N} x_i(t) = 1, \quad x_i(t) \geq 0 \quad \forall i$;

Obtain the solution of the sub-problem, $x_i^*(t)$;

Redirect $x_i^*(t)w(t)$ requests to server $i \quad \forall i$;

Update $q_i(t+1) \leftarrow [q_i(t) + x_i^*(t)w(t) - \delta_i]^+ \quad \forall i$;

Analysis of SEOL

We employ the Lyapunov optimization technique [71] which provides efficient and dynamic solution for stochastic optimization and control problems to derive SEOL as follows: first we define a Lyapunov function $L(t)$ measuring the total queue size in the cluster at time slot t. We choose $L(t) = \frac{1}{2}\sum_i q_i^2(t)$. Then, we define a Lyapunov drift as follows:

$$\Delta L(t) = E[L(t+1) - L(t) \mid \overline{q}(t)], \tag{6}$$

where $\bar{q}(t)=\{(q_1(t),q_2(t)),K,q_N(t)\}$. It is a well-established result [71] that minimizing the following function,

$$\min \Delta L(t)+\sum_{i=1}^{N} VE[\xi_i(t)] \tag{7}$$

provides a solution to our optimization problem. We need first find a bound for (7). Using $\max\{x,0\}^2 \le x^2$, one can show that the following inequality holds:

$$\begin{aligned} & q_i^2(t+1) \le (q_i(t)+x_i(t)w(t)-\delta_i)^2 = \\ & q_i^2(t)+\delta_i^2+x_i^2(t)w^2(t)+2q_i(t)x_i(t)w(t) \\ & -2x_i(t)w(t)\delta_i-2q_i(t)\delta_i \end{aligned} \tag{8}$$

Next, we use (8) to find a bound for the Lyapunov drift in (6). Note that we have:

$$\begin{aligned} & \frac{1}{2}\sum_{i=1}^{N}(q_i^2(t+1)-q_i^2(t)) \le \frac{1}{2}\sum_{i=1}^{N}(\delta_i^2+x_i^2(t)w^2(t)) \\ & +\sum_{i=1}^{N}(q_i(t)x_i(t)w(t) \\ & -x_i(t)w(t)\delta_i-q_i(t)\delta_i) \end{aligned} \tag{9}$$

The expectation of (9) given that $\bar{q}(t)$ yields the bound for Lyapunov drift in (6) as follow:

$$\begin{aligned} \Delta L(t) \le \frac{1}{2}E[\sum_{i=1}^{N}(x_i^2(t)w^2(t)+\delta_i^2-2q_i(t)\delta_i) \\ +\sum_{i=1}^{N}(q_i(t)x_i(t)w(t)-x_i(t)w(t)\delta_i)\,|\,\bar{q}(t)] \end{aligned} \tag{10}$$

Then, by simplifying the right-hand-side of inequality (10) we have:

$$\begin{aligned} \Delta L(t) \le B+\sum_{i=1}^{N}((\frac{1}{2}x_i^2(t)w^2(t) \\ +q_i(t)x_i(t)w(t)-x_i(t)w(t)\delta_i) \end{aligned} \tag{11}$$

Now, let δ^{max} be the maximum service rate over all the servers and $B(\delta^{max})^2/2$. We add our cost function $\sum_{i=1}^{N} VE[\xi_i(t)]$ to both sides of (11) and we have;

$$\begin{aligned} & \Delta L(t)+\sum_{i=1}^{N} VE[\xi_i(t)] \le B+\sum_{i=1}^{N}((\frac{1}{2}x_i^2(t)w^2(t) \\ & +q_i(t)x_i(t)w(t)-x_i(t)w(t)\delta_i)+\sum_{i=1}^{N} VE[\xi_i(t)] \end{aligned} \tag{12}$$

After rearranging (12) tt is easy to see that SEOL algorithm tries to minimize the right-hand-side of (12) which is basically the *drift plus penalty* technique developed in [71]. The idea behind SEOL is to minimize the right hand side of (12) at each slot and V is weight which can handle the trade off between energy usage and queue sizes (i.e. penalty and drift).

Solving the sub-problem in SEOL

In previous section we have derived SEOL algorithm but now SEOL requires the solution of a sub-problem in Algorithm 1 at each time slot. The sub-problem is a convex problem which can be solved analytically by applying Karush-Kuhn-Tucker (KKT) conditions and the optimal solution of the sub-

problem, $x_i^*(t)$, can be computed. We remove t index for notational simplicity, and then we derive Lagrangian function as follows:

$$(x_i, \lambda, \mu_i) = V.\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\frac{1}{2}x_i^2 w^2 + q_i x_i w - x_i w\delta_i - \lambda(\sum_{i=1}^{N}x_i - 1) - \sum_{i=1}^{N}\mu_i x_i \quad (13)$$

where $\lambda \text{and} \mu_i$, $\forall i$ are Lagrangian multipliers associated the first and second constraints, respectively in the sub-problem in Algorithm 1. Then, KKT conditions are given as,

$$\begin{aligned}
&\frac{\partial}{\partial x_i}(x_i, \lambda, \mu_i) = 0 && \forall i \\
&\frac{\partial}{\partial \lambda}(x_i, \lambda, \mu_i) = 0 && \\
&\frac{\partial}{\partial \mu_i}(x_i, \lambda, \mu_i) = 0 && \forall i \\
&\mu_i x_i = 0 \quad \text{and} \quad \mu_i \geq 0 && \forall i
\end{aligned} \quad (14)$$

The partial differential with respect to x_i is computed as,

$$\begin{aligned}
&\frac{\partial}{\partial x_i}(x_i, \lambda, \mu_i) = V\xi_i^{idle}\left(\frac{w}{c_i w + q_i}\right) \\
&+ V\frac{w(\xi_i^{peak} - \xi_i^{idle})}{\delta_i} \\
&+ q_i w - w\delta_i + x_i^* w^2 - \lambda^* - \mu_i^*
\end{aligned} \quad (15)$$

Eventually the following system of equations should be solved in order to find the x_i^*. We define F_i such that:

$$F_i \equiv V\xi_i^{idle}\left(\frac{w}{c_i w + q_i}\right) + V\frac{w(\xi_i^{peak} - \xi_i^{idle})}{\delta_i} q_i w - w\delta_i$$

Then, the system of equations is given by,

$$\begin{aligned}
&F_i + x_i^* w^2 - \lambda^* - \mu_i^* = 0 \quad \forall i \\
&x_1^* + x_2^* + \ldots + x_N^* - 1 = 0 \\
&\mu_i^* x_i^* = 0 \quad \forall i
\end{aligned} \quad (16)$$

This problem has basically the well-known water-filling solution. After solving the system, x_i^* is obtained as follows;

$$x_i^* = \begin{cases} \dfrac{\lambda^* - F_i}{w^2} & \text{if } \lambda^* > F_i \\ 0 & \text{if } \lambda^* \leq F_i \end{cases}$$

In more compressed form, $x_i^* = [\frac{\lambda^* - F_i}{w^2}]^+$. Also we know that the sum of all x_i^* should be one. That is:

$$\sum_{i=1}^{N} [\frac{\lambda^* - F_i}{w^2}]^+ = 1 \quad (17)$$

In equation (17), we need to compute λ^*. As long as (17) is monotone and increasing, it is possible to use iterative methods in order to find λ^*. We inspired from [72] and derive iterative method given in Algorithm 2 to achieve λ^*.

Algorithm 2: Iterative Algorithm

Input: Total request rates: (w).
Vector F: (i.e. $\vec{F}$)
Output: The optimum value of λ^*

1: $g(\lambda) \triangleq \sum_{i=1}^{N} [\frac{\lambda - F_i}{w^2}]^+ - 1$, $\vec{F'} \leftarrow \text{Sort}(\vec{F}, "ASC")$
2: $N' \leftarrow N$, $F'_{N'+1} \leftarrow \infty$
3: **while** $N' > 1$ **do**
4: **if** $F'_{N'} < F'_{N'+1}$ and $g(F'_{N'}) < 0$ **then**
5: $L \leftarrow F'_{N'}$
6: $U \leftarrow F'_{N'+1}$
7: **break loop**
8: **else**
9: $N' \leftarrow N' - 1$
10: **end if**
11: **end while**
12: $\lambda^* \leftarrow (L + U)/2$
13: $\epsilon \leftarrow |g(\lambda^*)|$
14: **while** $\epsilon > 10^{-7}$ **do**
15: **if** $g(L) * g(U) < 0$ **then**
16: $U \leftarrow \lambda^*$
17: **else**
18: $L \leftarrow \lambda^*$
19: **end if**
20: $\lambda^* \leftarrow (L + U)/2$
21: $\epsilon \leftarrow |g(\lambda^*)|$
22: **end while**

The above algorithm consists of two parts (i.e. two iterative parts) which lead to finding λ^* and the corresponding $x_1^*, x_2^*, \ldots, x_N^*$ can be computed from (16). The first loop has a limited number of iterations (worst-case N iterations) and tries to find bounds L and U such that $L < g(\lambda) < U$. In order to see a proof of convergence of similar algorithms, we refer readers to [17]. In the second part of Algorithm 2 iteratively we try to find λ such that $g(\lambda) = 0$ by using the bounds found in the first parts (i.e. L and U). Wee use the bisection method and the bisection iterations stop when the error is significantly small (we set less than 10^{-7} in our case).

3.4.4 Simulation results

The performance of the SEOL algorithm is evaluated. In order to compare our stable energy optimal load balancing (i.e. SEOL) algorithm, we implemented two other commercial load balancing algorithms [73], briefly introduced here. Algorithm RAND redirects all incoming traffic $w(t)$ to one server which is randomly chosen (i.e. with uniform distribution) in each time slot and does the same in the next time slot and so on. We also implemented *Round Robin* (RR) which assigns all incoming traffic (i.e. $w(t)$) to one server in circular order. These algorithms are static so far, therefore, more dynamic methods are also considered. The 2RC algorithm selects two servers randomly and redirects all traffic to the one with lower load (queue length is considered as load) in each time slot. In more dynamic schemes,

the *Least Loaded* (LL) algorithm redirects all traffic to the server with the shortest queue length. However, all these algorithms are oblivious to energy usage and in order to be comparable with our energy-aware SEOL algorithm, we implemented a sleep on/off mechanism so that all servers with an empty queue can go into sleep mode in each time slot. Finally, we consider one extreme case of LL scheme (we named it individual Least Loaded or iLL) which redirects requests one by one, in other words, for each request in each time slot, it tries to find the server with minimum load and assign the current request to that server, this scheme can achieve very good load balancing and stability with low queue length but practically is very challenging to implement because always should track the servers queue backlogs and we use it as best possible benchmark in load balancing.

Here, we define our platform and simulation configuration which was the same for all algorithms. The platform is a cluster of servers (e.g. a datacenter) consisting of 100 servers and a load balancer. We considered the same service rate with $\sigma_i = 100 \forall i$ requests per time slot and the duration of each time slot is set to $\tau = 2s$). We considered four different types of servers in terms of power usage $P^{idle} = 50, 60, 70 \text{ and } 80w$ and $P^{peak} = 100, 110, 120 \text{ and } 130w$ and there were 25 servers of each type. We use a Poisson arrival process in three scenarios with stationary mean values 2000, 5000 and 9000 requests per time slot. The simulation time was 100 thousand time slots. We assigned c values in the range $[1..100] * 10^5$ and gave the higher numbers to servers with lower energy usage. We tested several scenarios with different average arrivals for all algorithms and different V values.

The first result we present is the energy usage of all scenarios. Figure 53 shows the time average energy usage of the whole cluster by different V values for SEOL and three average arrivals. As the other algorithms (i.e. LL, 2RC, RAND , RR) redirect all traffic to one server, they have almost similar energy usage,therefore, we only show LL (which has the lowest energy usage among them) with iLL and SEOL. As can be seen in Figure 53, for higher V values we can save more energy using our algorithm and for the case with average 9000 requests in each time slot, convergence to the optimum point can be achieved with lower V values because with high arrival rates (e.g. near to full capacity of the cluster 10000 requests per time slots), there is not much room to trade off between energy and stability. Obviously, iLL does not perform well because it keeps all servers active all the time and therefore shows the upper bound of energy usage. For the case with lower average arrival rate (e.g 2000), we can save more energy while maintaining stability.

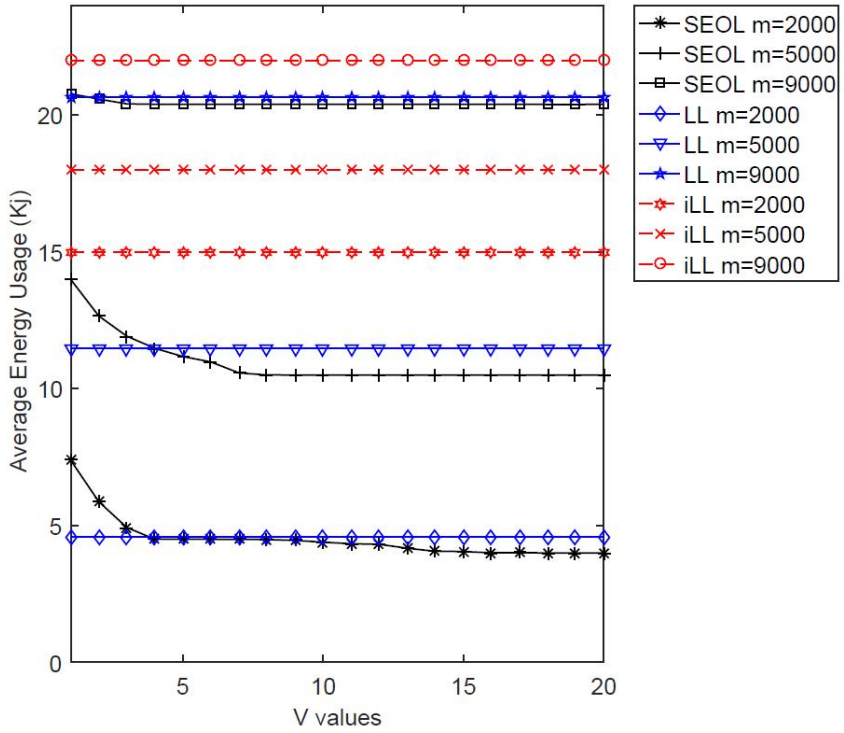


Figure 53: The energy usage by the clusters with SEOL and LL algorithms

Next, we show results from a stability analysis where we plot the overall queue backlogs of all servers over time. Here we ran the simulation with all algorithms with mean arrival rate of 9000 requests per time slot and we selected three V values for SEOL. Although higher V values give us better energy saving, the penalty is increased convergence time to reach stable state and the queue backlog is also higher. Moreover, Figure 54 shows that SOEL performs with lower queue lengths compare to all other algorithms apart from iLL which is considered as best case (in our scenarios) in convergence time and queue lengths. It is worth noting that LL and RR perform the same and both curves are identical. The figure also shows how lower V values can give lower total queue length and faster convergence.

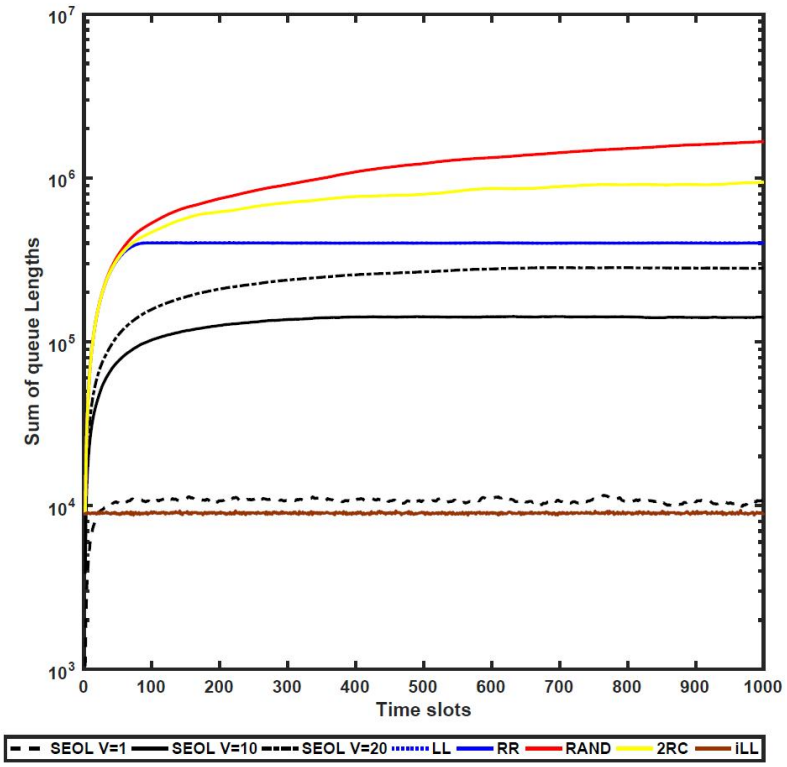


Figure 54: The stability of the cluster during time

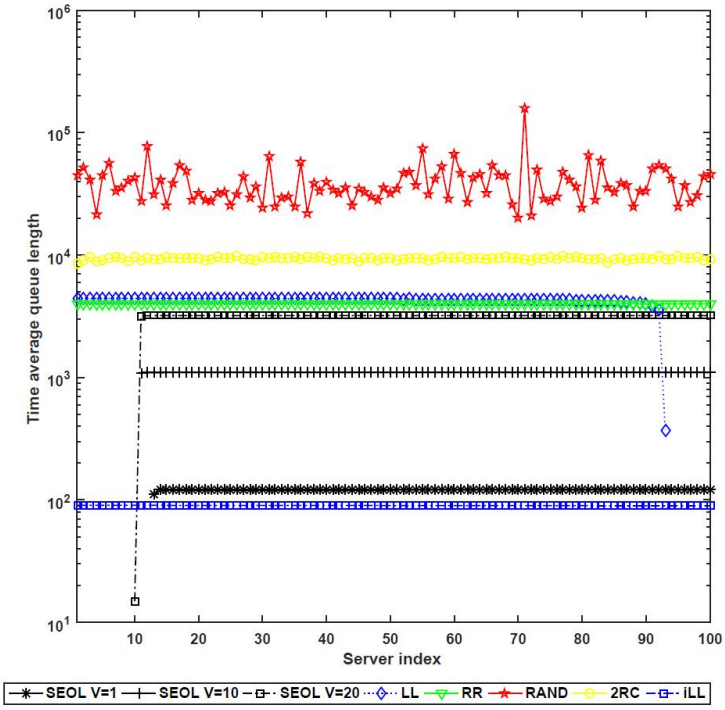


Figure 55: The time average load of each server

Finally, we are also interested in load balancing. We define load balancing as average queue length per server and Figure 55 shows the performance of all algorithms. In this setup, we considered traffic with 9000 arrivals on average and one can see SEOL and LL use about 90 percent of all servers during the time (because 9000 is on average 90 percent of the total service rate of the cluster of 100 servers with service rate 100 each) but SEOL achieves lower load. In the other algorithms (i.e. RAND,

2RC and RR) all servers have something to serve. The RAND performs with highest variation and SEOL balances the load nicely with very low variation. iLL achieves the lowest queue lengths and uses all servers as expected, and can be considered as a lower bound in this case. Again, the lower V values brings shorter queue lengths in the spatial case (i.e. different servers) as it did nicely in the temporal case (i.e. different time slots). As a result, we highlight that our algorithm is flexible in achieving a desired trade-off between server load stability and energy usage. Moreover, in many scenarios (especially when the arrival is much lower than capacity of the system), if the right V value is selected, it is possible to save significant energy by sacrificing a little response time. As an example, if we have arrival with average 5000 requests per time slot, iLL performs with average 27 percent lower queue lengths in load balancing compare to SEOL $V=4$ which can save 47 percent energy. In another example with low arrival rate (i.e. 2000 requests per time slot), the freedom to switch more servers to sleep mode is higher as mentioned before, therefore, iLL performs with 10 percent lower queue lengths compare to SEOL $V=4$ which performs with 70 lower energy usage.

3.4.5 Conclusions

In this study, we have addressed the challenge of load dispatching in data centres with the requirement of preserving time-varying queue stability and saving of energy-usage, simultaneously. To this end, we have developed a stochastic optimization problem, and derived an algorithm analytically. We show that our proposed algorithm, SEOL, can simultaneously reduce energy usage and queue length of servers. Furthermore, SEOL can offer a desired trade-off between queue length and energy usage by properly tuning a single parameter (V). Specifically, we showed that when the data centre load is high (but below the data centre capacity), setting V to small values leads to shorter queue length compared to the existing load balancing techniques. For high values of the tuning parameter V , a substantially higher gain of energy saving is achieved while the queue stability is also preserved. A potential future work could be focused on extending the analysis to obtain an optimality bound for our proposed algorithm. Also, more heterogeneous cases with different server capacities and non-stationary job arrivals could be addressed.

3.5 Fair and efficient resource sharing in interconnected CDNs

3.5.1 Introduction

The current solution to the worldwide transfer of massive traffic is commercial Content Delivery Networks (CDNs). This traffic is outsourced by content providers (CPs) to commercial CDNs, and traverses the network in the form of over-the-top (OTT) traffic. The lack of control from network operators over this traffic has caused increased congestion on their networks and degraded quality of experience (QoE) of their subscribers. Despite these strains, network operators are marginalized in the revenue chain of content delivery which is currently monopolized by few commercial CDN owners. This has led the network operators to find strong motivations to deploy and run their own content delivery infrastructures, a phenomenon that has led to the rise of *Telco CDNs* [74] [63] [75]. Beyond the monetary profit gained by being involved in the revenue chain and an enhanced control over the traffic matrix, a Telco CDN is inherently suitable to manage and deliver User Generated Content (UGC) due to the following reasons: first, the trajectory of a UGC is bottom-up, meaning that a UGC is generated and uploaded by a subscriber in the network operator domain. Second, UGCs exhibit strong locality attributes, evidenced by the recent studies on social networks showing that approximately 84% of the social community of a user in Facebook is collocated in the user's residence country [64]. This implies that a substantial fraction of the total demand for a UGC is likely to come from the subscribers within the footprint of the network operator and from its neighbors. By serving this demand locally, the network operator reduces its costs by reducing the amount of traffic traversing the expensive transit links charged by top tier transit providers. Besides, the operator will be able to enforce its traffic engineering policies to minimize congestion costs and improve the QoE of subscribers. Beyond UGCs, one can think of a plethora of other (commercial) contents with regional usage property, giving rise to the operator's profit if they are managed and delivered locally.

A Telco CDN offers vertical merits by providing the operator with a control over both traffic engineering and content distribution, and with a flexibility of jointly optimizing content delivery cost and QoE [76] [77]. However, its horizontal merits are limited because it is bound to an exclusive service region limited by the footprint of the operator's network [78]. This leads to a walled garden content delivery system, limiting the economy of scale. To counteract these limitations faced by Telco CDNs, the Internet Engineering Task Force (IETF) has proposed a framework termed *CDN Interconnection* (CDNI) [79]. CDNI specifies the architectural and signaling aspects. However, it does not address the intelligence required by the content management systems (CMSs) of the individual CDNs engaging in an interconnection. A CMS, as a central element of a CDN, implements content placement and demand assignment (or routing) policies [80]. It is safe to claim that the outcome efficiency of a Telco CDN is greatly driven by its CMS. As such, the efficiency of a Telco CDN interconnected with other CDNs is driven by the strategies chosen by its CMS in reaction to the strategies of other CMSs. However, this requires additional intelligence to be devised in a CMS in order to efficiently operate in orchestration with other CMSs. CMS orchestration is realized by implicit or explicit coordination of content management strategies among participating CMSs. It involves the endowment of resources (e.g. bandwidth, processing, and storage) among the CMSs; however, the decisions on the mutual endowments are governed by the rationality of the CMSs, which individually seek cost minimization and QoE enhancement. In the view of this, in this section we provide an efficient and fair sharing of resources among interconnected CDNs owned and operated by ISPS.

3.5.2 System Modelling

Suppose there are a set of ISPs $I = \{0, 1, \dots, N\}$, with ISP0 representing an ISP having a set of content in custody and wanting to share its surplus resource R among other ISPs as the customers of those contents. The resource R can be link bandwidth or CPU processing that is endowed by ISP0 to customer ISPs to deliver demanded content to their local users. The goal is to efficiently share this resource, taking into consideration that these ISPs would have experienced different costs if they wanted to serve the content locally by relying on their own resources. Alternatively, the customer ISPs would gain different utilities if instead the use the endowed resource of ISPs to serve their local demands. Formally speaking, ISP0 must determine a vector $\boldsymbol{r}^* = (r_1, r_2, \dots, r_N)$ such that $\sum_{i=1}^{N} r_i = R$ and $\boldsymbol{r}^*$ is an efficient sharing of resources.

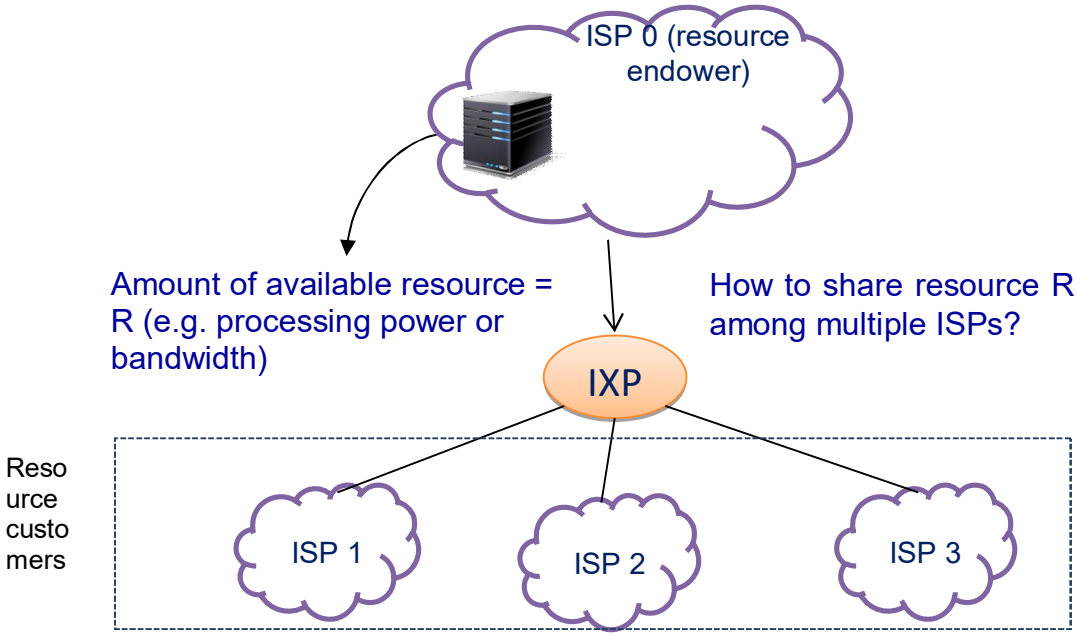


Figure 56: Topology of interconnected ISPs as Telco CDNs. ISP0 is resource endower to customer ISPs 1, 2 and 3.

We assume a general utility function represented as $\boldsymbol{u_i(r_i) = a_i(r_i)^{b_i}}$ where $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ are cost factors of the i[th] ISP (these could be energy cost, storage cost, bandwidth cost, etc., all combined in a single well-formed function). This function is borrowed from a recent INFOCOM paper:

- Hasan, Syed, et al. "Trade-offs in optimizing the cache deployments of CDNs." *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014.

It also conforms to the following studies:

- W. B. Norton, "The Internet Peering Playbook: Connecting to the Core of the Internet," *DrPeering Press*, 2012.
- I. Castro and S. Gorinsky, "T4P: Hybrid Interconnection for Cost Reduction," in *Proceedings of NetEcon*, 2012.
- I. Castro, R. Stanojevic, and S. Gorinsky, "Using Tuangou to Reduce IP Transit Costs," *IEEE/ACM Transactions on Networking*, 2014.

This parsimonious representation of cost (e.g. for energy consumption) greatly simplifies our further analysis, however any other function can be applied in our model

In the sequel, we introduce three different strategies of sharing resource R among customer ISPs. For the simplicity of our discussion, we consider a set of four ISPs as illustrated in Figure 56.

Global maximization of cost saving (aka. Social Optimal)

- $\boldsymbol{(r_1^*, r_2^*, r_3^*) = \underset{(r_1,r_2,r_3)}{\mathbf{argmax}} \{u_1(r_1) + u_2(r_2) + u_3(r_3)\}}$
- Subject to $\boldsymbol{r_1 + r_2 + r_3 = R}$

Uniform resource sharing

- $\boldsymbol{r_1 = r_2 = r_3 = \frac{R}{3}}$

Nash Bargaining Solution (NBS)

- $\boldsymbol{(r_1^*, r_2^*, r_3^*) = \underset{(r_1,r_2,r_3)}{\mathbf{argmax}} \{u_1(r_1) * u_2(r_2) * u_3(r_3)\}}$
- $\boldsymbol{r_1 + r_2 + r_3 = R}$

3.5.3 Numerical Results

In this section, we compare the cost saved by using the three strategies mentioned in previous section, namely, social optimal, uniform allocation, and NBS. We use the ISP topology demonstrated in Figure 56. Without loss of generality, we assume in the utility function $a_i = 1$ for all ISPs. Thus, the ISP utilities differ by the cost exponent b_i. In the first experiment (shown in Figure 57 and Figure 58) we assume that the ISP0 wish to share its entire resource R, whereas in the second experiment (shown in) we assume ISP0 wish to spare a fraction of its surplus resource (not to share it) if that makes the total cost saving better. The vector of cost factors is illustrated on top of the figures. The indexes of this vector are equivalent to ISP indexes, sorted from 1 to 3 (for the first experiment) and 0 to 3 for the second one. For illustration purpose, we assume the resource subject to sharing is bandwidth.

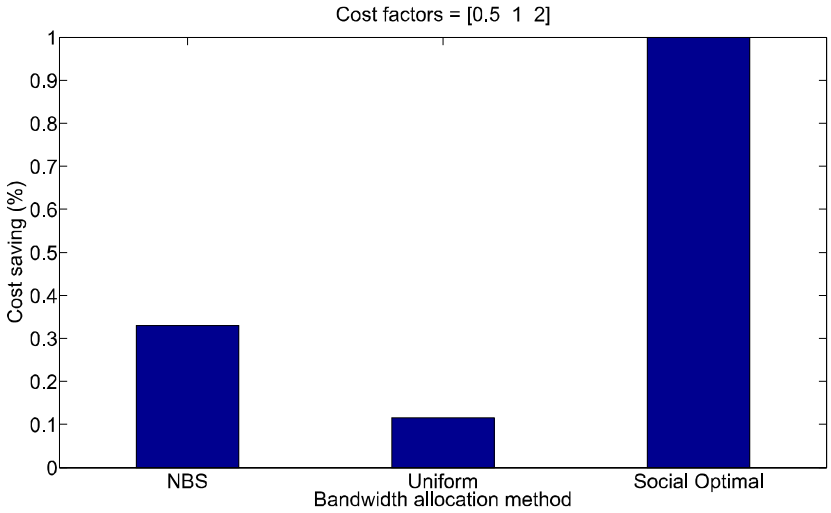


Figure 57: Comparison of total cost saving using three different strategies. ISP0 is pure endower and ISPs 1, 2 and 3 are resource customers with cost vector [0.5 1 2]

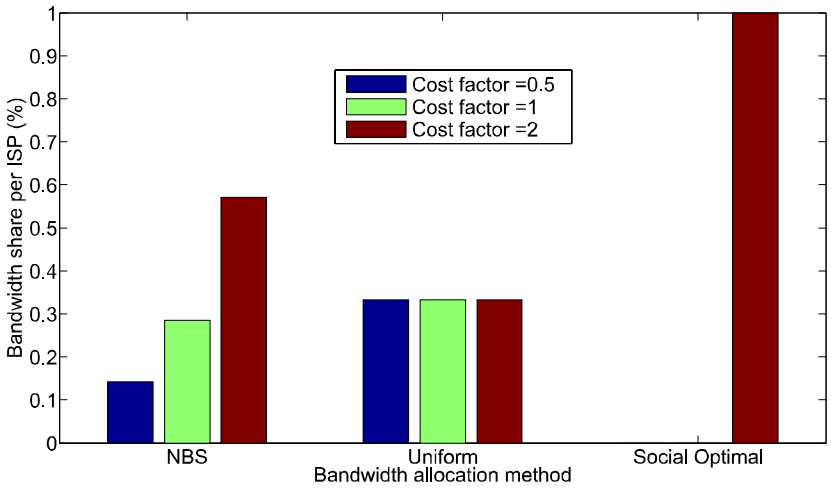


Figure 58: Comparison of bandwidth sharing among customer ISPs using three different strategies. ISP0 is pure endower and ISPs 1, 2 and 3 are resource customers with cost vector [0.5 1 2]

Figure 57 indicates that the social optimal archives the largest cost saving, then NBS, and uniform allocation as the worst. On the other hand, as observed in Figure 58, the social optimal yields the worst fairness, while the uniform allocation offers maximum fairness. NBS archives a trade-off between cost saving and fairness, a property that cannot be achieved by other strategies.

Similar behaviour can be observed in Figure 59 and Figure 60 when ISP0 is allowed to keep some part of its surplus resource unshared. Again, one can observe that NBS achieves a good trade-off between cost saving and fairness.

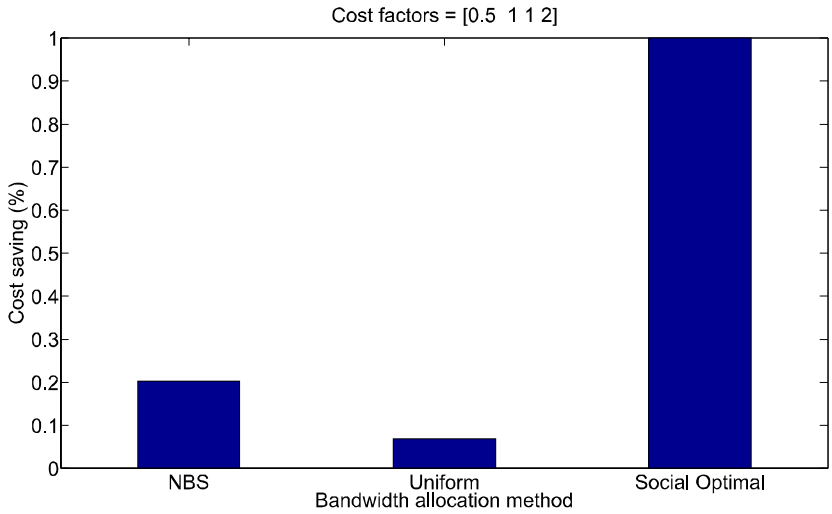


Figure 59: Comparison of total cost saving using three different strategies. ISP0 is resource endower, but it can keep some resources unshared. ISPs 1, 2 and 3 are resource customers. The cost vector for the 4 ISPs is [0.5 1 1 2]

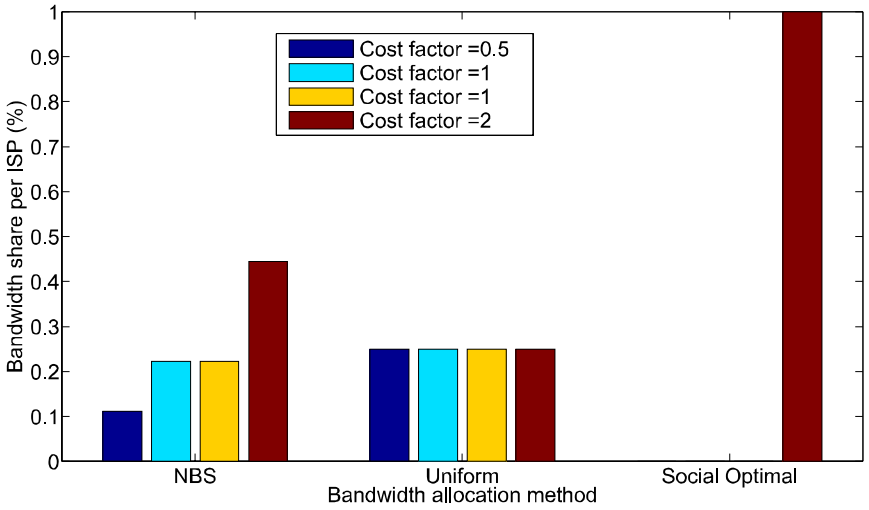


Figure 60: Comparison of bandwidth sharing among ISPs. using three different strategies. ISP0 is resource endower, but it can keep some resources unshared. ISPs 1, 2 and 3 are resource customers. The cost vector for the 4 ISPs is [0.5 1 1 2]

3.5.4 Conclusions

Our study of cost (or utility) sharing among multiple ISPs participating in content distribution reveals that efficient mechanisms exist to reach a trade-off between the total cost saved and the share of resources (e.g. bandwidth) allocated to individual ISPs. In particular, we showed that NBS has a desirable potential to achieve such a trade-off. In addition, NBS (or similar bargaining techniques) is simple and parsimonious which makes it easy to implement as a third party component responsible for resource sharing between ISPs. This approach can be employed in the IETF CDN Interworking (CDNI) framework to support efficient interworking among multiple (Telco) CDNs.

3.6 Resource-aware distributed content delivery

In this section we present mechanisms for optimizing energy consumption in video delivery to mobile devices. Our focus is on distributed mechanisms for Web-based video delivery, as the emerging WebRTC technology allows direct communication between modern web browsers for streaming video and transmitting arbitrary data. We present a design of resource-aware distributed content delivery based on WebRTC, which enables building scalable and energy-efficient video delivery systems for mobile Web.

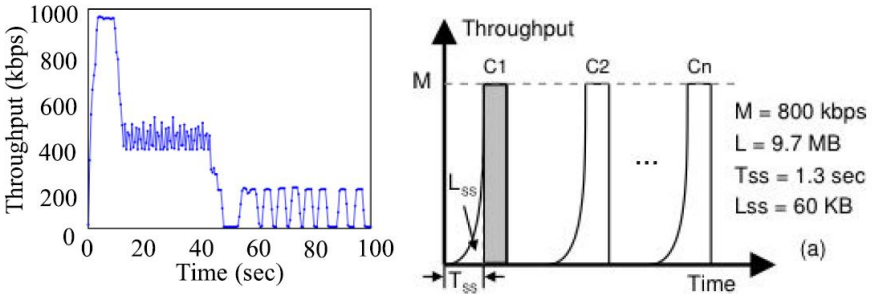
3.6.1 State-of-the-art

There is a fair amount of research about energy-efficient video streaming and delivery solutions for mobile devices in general. However, in the context of mobile Web, the research related to energy-efficient video content delivery seems to be quite a new topic.

Energy-efficient video delivery solutions

Hoque et al. [81], [82] examine solutions that have been proposed during the last few years, to improve the energy efficiency of wireless multimedia streaming in mobile hand-held devices. They categorize the solutions based on the different levels of the Internet protocol stack, and also based on different scheduling and content adaptation mechanisms. Their findings indicate that some energy saving mechanisms work well together but there are also conflicting mechanisms that actually decrease the energy efficiency when used in conjunction with each other. The scheduling and content adaptation mechanisms are the most relevant ones in the scope of this document.

The presented scheduling mechanisms aim to transfer the video data in bursts, with the purpose that the radio interface can enter low-power state between the bursts. An example of this type of scheduling mechanism can be seen in Figure 61, which shapes the traffic pattern of YouTube video stream into distinct bursts transferred at the maximum download speed of the receiving device.



(a) YouTube Traffic Pattern (b) YouTube *chunk-mode* traffic pattern

Figure 61: Comparison between YouTube regular traffic pattern and proposed chunk-mode traffic pattern [81]

The content adaptation mechanisms can be divided in three categories: scalable video coding (SVC), transcoding and content selection. SVC provides the capability to code a single video stream into multiple layers of different bit rates and quality levels. The base layer corresponds to the lowest bit rate stream having the minimum quality, frame rate and spatial resolution. The enhancement layers increase the quality of the stream by increasing the frame rate and spatial resolution. The number of layers to be transmitted to a streaming client at any time is determined by a flow control algorithm based on the feedback received from the client. Therefore, this technique has potential to reduce network traffic and computational complexity at the mobile devices, which in turn reduce power consumption.

Transcoding is another way to adopt the video content based on different application, device or network requirements. In this approach, only one bit stream of high quality video is stored at the server. At some part of the network, this high quality video stream is transcoded into lower quality versions that fulfil the requirements of the different users.

Content selection is similar to transcoding as it provides different quality versions of the same video content to the client. However, in this approach multiple copies of the same video content at different quality levels are stored at the server, which increases the storage requirements.

Distributed video delivery system architectures

The effect of the video delivery system architecture to energy efficiency has also been studied. Three main categories for building the system architecture have been identified: data center, peer-to-peer (P2P), and hybrid architectures.

The energy consumption analysis provided in [83] shows that there is no significant difference in the total energy consumption of a video streaming system between a data center based and a P2P based architecture. This is illustrated in Figure 62. However, the authors state that P2P based architecture will become a green alternative to data centers as end-user devices and routers become more energy efficient.

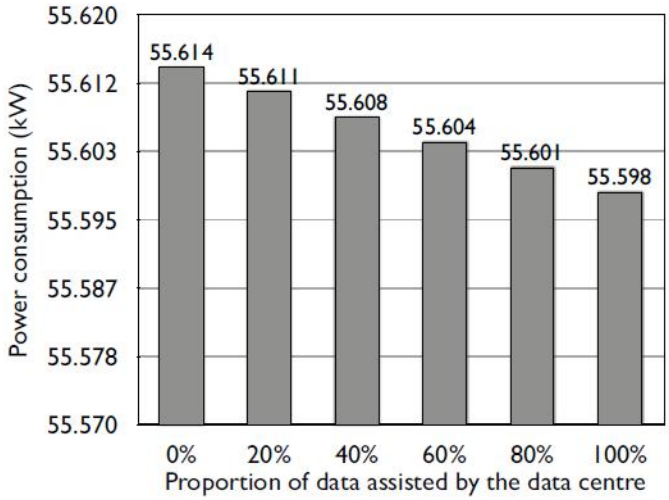


Figure 62: Overall power consumption of a video streaming system when moving from pure P2P solution towards data center based architecture [83]

In [84], energy consumption models of video-on-demand (VoD) services with different delivery methods are presented. The study presents simulated energy consumption results for proposed localized hybrid peer-to-peer (HP2P) and localized peer-assisted patching (PAP) with multicast video delivery methods over optical access networks, which are compared against a conventional CDN solution. The results show that the popularity/demand of the content is an important factor for selecting the optimal video delivery method.

Both of the proposed methods utilizing P2P are shown to outperform the traditional CDN regarding energy efficiency, as shown in Figure 63. Localized PAP with multicast is shown to be the most energy efficient when delivering popular VoD content. Localized HP2P is the most energy efficient with less popular VoD content.

Video delivery in the mobile Web

Web-based video consumption has become very popular and makes for a significant portion of the total Internet traffic today. In addition, with the rapid development and popularity of capable mobile devices such as smartphones, more and more video content is consumed in a mobile Web context. However, energy efficiency in this domain has not yet been widely studied [85].

HTTP based video delivery methods have been developed, that can utilize the existing CDN infrastructure [86]. In addition, P2P technology has become available also to the mobile Web browsers with emerging WebRTC API. Preliminary research results show that WebRTC is a promising technology for distributing video delivery in the Web domain [87].

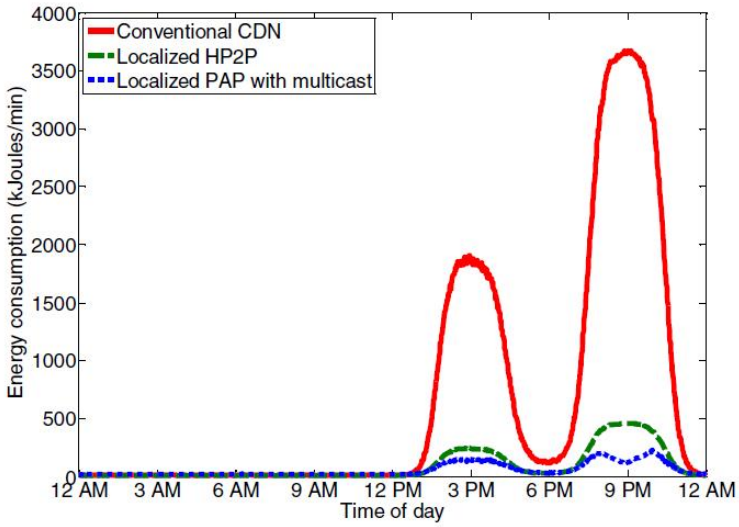


Figure 63: Video delivery network energy consumption in a day for popular VoD channels [84]

3.6.2 A design of resource-aware distributed video delivery

Our Web-based distributed video delivery solution is based on a hybrid architecture, where in addition to the traditional content server(s) or CDN, video content is transmitted also from other users using WebRTC technology. Video data is divided into smaller segments, which can be downloaded in parallel from both servers and other peers. Resource-awareness is utilized for selecting the optimal nodes for downloading the data in order to optimize performance and energy consumption. An overview of the proposed architecture can be seen in Figure 64.

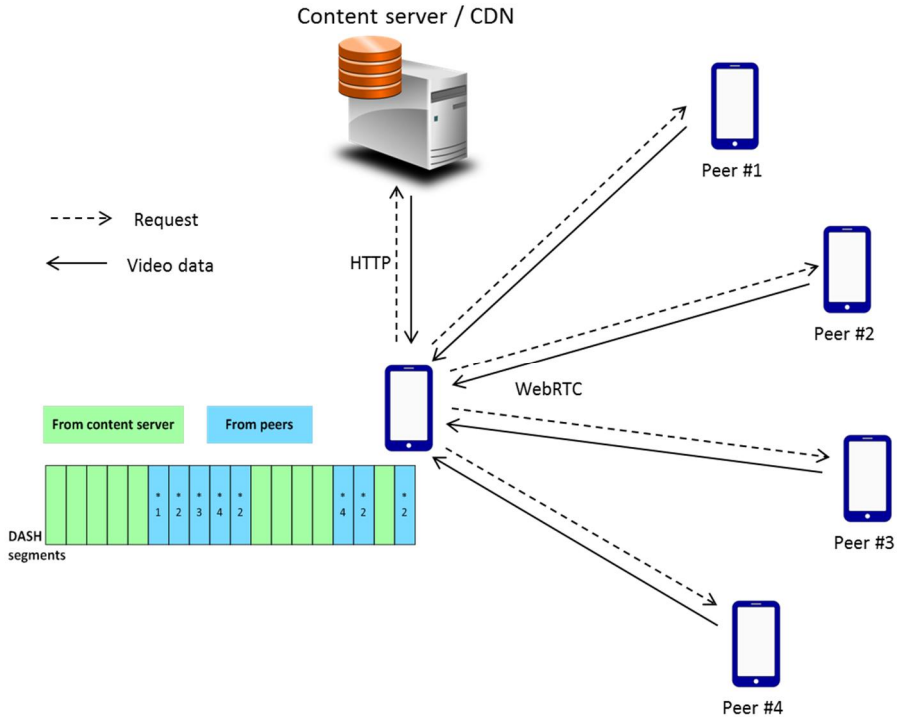


Figure 64: Overview of the resource-aware distributed video delivery architecture

We utilize MPEG-DASH for dividing the video data into smaller segments that can be downloaded individually. This allows controlled downloading of the video data simultaneously from multiple

sources. It also allows selecting the quality level of each segment individually from multiple different quality levels available in the system, based e.g. on the current mobile signal strength. The segments are downloaded from the server side using HTTP and from the other peers using WebRTC. The selection of download sources is done in a resource-aware manner, where e.g. the peers' network bandwidth and battery level are taken into account. This allows optimizing the performance e.g. in order to minimize download times. Video buffering is done in bursts in order to allow the mobile radio interface to enter low-power state between the bursts.

3.7 Leveraging social information to enhance video delivery

3.7.1 Introduction

With the eruption of Online Social Networking (OSN) there exist a large number of unpopular content, e.g., User Generated Content (UGC), which, although hosted in content providers, is actually shared and accessed through OSNs. According to the statistics of Alexa ranking the second and third most traffic generated websites in the Web are Facebook and YouTube respectively. Moreover, Google sites (i.e., Youtube.com) are considered as the largest video streaming platform whereas Facebook is able to reserve the second place in accordance with the analysis of U.S. desktop online video rankings. In this era, sharing Video content is increasingly popular among users in OSN. This activity includes sharing both user generated content or other interesting videos generated by other social connections, which may forward flood of requests to content providers across social links This is a very costly and complex task to host, manage and deliver a huge number of content that popular social networks should deal with them every day. Therefore most of them rely on external major CDNs to distribute their content from remote servers to edge locations (e.g., Facebook uses Akamai) by providing on-demand services or live streaming to users while satisfying QoS. One of the main advantages over CDNs is that, it allows users to experience less jitter for streaming applications by placing edge servers nearby. Even though content providers use CDNs to distribute most popular content globally, high setup fees and other hidden cost may associate with CDN vendors. This study aims to propose a new architecture which is capable of distributing unpopular content but locally popular among particular user group identified from their social information.
This allows content providers to distribute their content with low cost in specified geographical regions. Apart from that, many studies have been conducted to optimize multimedia content delivery in CDNs such as optimizing energy consumption of the CDN network, improving caching techniques, and content delivery optimization based on social graphs. Usually content providers are very efficient on serving popular content since they can easily predict and identify locality and temporal patterns associated with them. This allows CDNs to improve QoS and reduce delivery costs of the content providers. However, prediction of unpopular UGC is difficult due to lack of information about when and from where these contents available. That means, end user needs to retrieve UGC content from central facilities hosted by the content provider (i.e., datacenters), which may incur in long paths, which are harmful for both end user (i.e., long access delays) and content provider due to proper resource (e.g. bandwidth) allocation is necessary to serve the query. This may not be an issue for a single or few queries, but for large content providers like YouTube may generate millions of queries only for few visits of different videos. Thus, delivering such a video content via CDNs is inefficient and costly. A potential solution to alleviate this problem would be to use the device of an end user as cache of some content, so that they can deliver (totally or partially) the content under some circumstances. We can find previous proposals in the literature that propose using peer-assisted content delivery. For instance, the authors of [88] evaluate the benefits and risk of using peer-assisted Akamai CDN. They conclude that using peers' uplink to serve pieces of the content helps to offload from 60% to 80% of the traffic to end users without suffering quality degradation. Although these proposals are very interesting to demonstrate peer-assisted content delivery in research community as well as in commercial solutions, the main focus is based on relatively popular content. All of them assume that there are some users that have cached partially or completely the content they are helping to serve. Therefore, all the end nodes caching the content form a swarm that alleviates central servers. Still, we believe that none of these proposals would work for actually unpopular content that receives at most tens of visits in a day. For those cases we propose a social-peer-assisted content delivery approach (namely SocialiVideo) whenever it is possible. This approach is based on the users' social information specially their social connections (friendship) and location information. Integration of users' geographical location with OSNs allows opportunities to enhance content delivery on CDNs as elaborated on this study.
In this study, we propose an idea of delivering (rather) unpopular content shared among end users on an OSN. The idea is to disseminate video content among users in social networks based on users' location obtained from OSN information. If user shares videos in social networks, connection request is forwarded to one of the CDN edge servers. Then the response for a particular request directs via one or many hops based on the content availability of the servers. We propose an idea to optimize this process by placing content very closer to users using their local resources (e.g., home set-top box with 24/7 functionality). Once a user pulls the video content from the server, the other users closer to him can request the same content contacting the edge server or friend's local server. This helps to reduce bandwidth consumption of the network. Furthermore, if two users are located in the same network,

offline download can be performed for available content. This simple but efficient way of content delivery mechanism can reduce the load of the social network CDNs as well as the overall traffic load.

3.7.2 System Architecture of SocialiVideo

Figure 65 illustrates the functional view and data traffic flow of the proposed SocialiVideo (SiV) solution which aims to enhance Facebook content delivery. The proposed SiV method is more elaborated below based on three algorithms and two modules; LSiV (Local SocialiVideo) and CSiV (Central SocialiVideo). CSiV is a central server that keeps track of the users profile info (location, social relationships, etc) and metadata of the disseminated content such as URL of the shared videos. In-order to perform load balancing, CSiV facilitates to keep the video URL of both local storage and Akamai edge server.

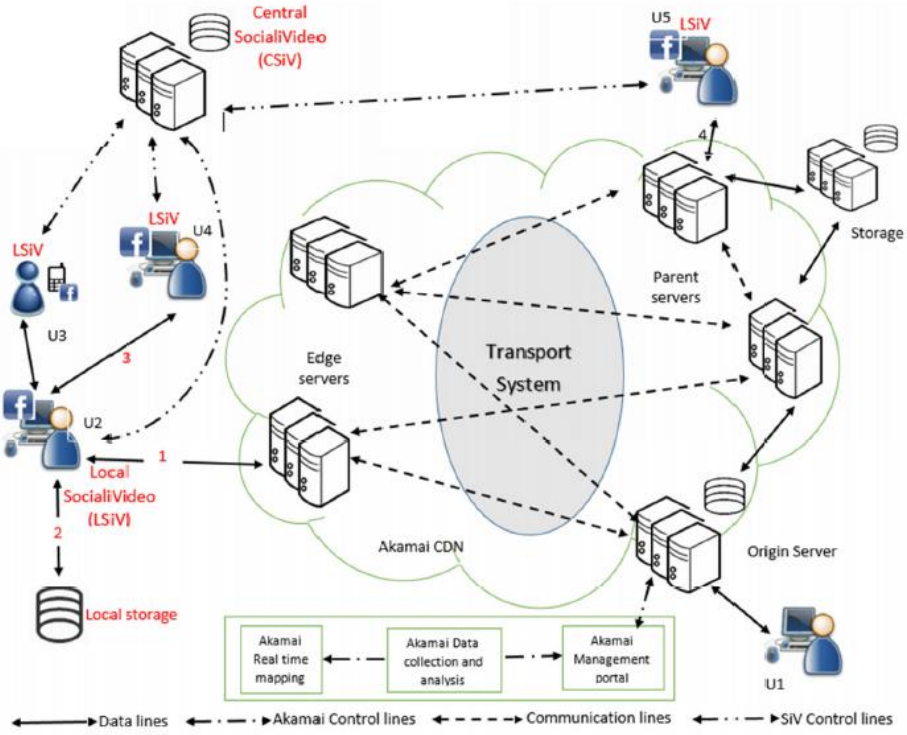


Figure 65 - System architecture of SiV

LSiV keeps track of information pertaining to each single user in the client side. Data synchronization is performed between LSiV and CSiV whenever updates are available. Additionally, three algorithm functionalities are as follow. Algorithm 1 (algorithms are presented on initial version of the deliverable) execute when user upload or share a video link in the FB. When another user tries to stream the shared video of one of his friends FB, Algorithm 2 evaluates the possibilities. Algorithm 3 checks when to perform cache optimization in local storage. In each scenario, we assume that each user executes SiV algorithm on his device.

3.7.3 Analysis of SocialiVideo

This part differentiates few main characteristics of SiV with the FB/AK approach which presented completely in the initial version of this deliverable. Our solution architecture is based on the FB/AK approach, but SiV uses P2P distributed services. Session path and number of hops from the end users also important, which is higher in the FB/AK due to forwarding queries to the edge server clusters placed in ISPs/PoPs. In addition, SiV solution works as an ISP friendly service (in cases that users are in the same network/city) as the content is in users' local server and also, it reduces inter ISP traffic due to P2P streaming. Another point is capital intensive, which in SiV is much lower as it uses existing user premises and the existing underline network. However, implementing CDN replica server clusters in different places is very costly. Our presented analysis in following sections show that SiV solution is reducing the global propagation delay, which is roughly 100 ms for a one way interactions in a usual session [89]. Lastly, both FB/AK and SiV use HTTP flows to deliver streaming

content as a reliable transmission. More information about the energy, latency, and traffic parameters is given in the following sections.

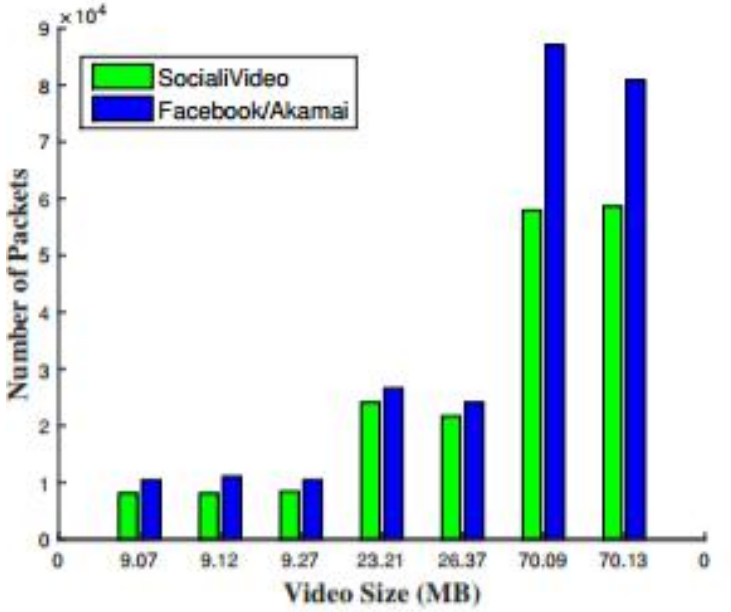


Figure 66 - Number of delivered packets

The SiV solution aims to reduce power consumption of network devices while minimizing the traffic load on the intermediate devices by serving video content in a P2P fashion between closer users. To evaluate the enhancement that this solution can offer, we implemented a prototype and collected a sets of data and attributes (e.g. number of packets, number of hops, transmission time, and delay). Our test bed consists of three main components. i) CSiV-hosted in Heroku cloud application platform ii) LSiV-configured in a local computer iii) SiV user-in the same network as LSiV. We evaluate the performance by using 7 HD quality videos of different lengths based on two scenarios; i) When user using SiV ii) When user is not using SiV, but regular FB/AK content delivery mechanism. Figure 66-Figure **67** illustrate three conducted experiments in this regard. The first experiment in Figure 66 shows number of packets delivered in seven different HD videos that were popular during the data collection period. We fetched videos in both FB/AK and SiV solutions separately, 10 times per day (from 10:00 to 14:00) for a duration of a full week. As the figure shows, for all the video delivery, FB/AK approach uses slightly more packets compared to the proposed SiV solution.

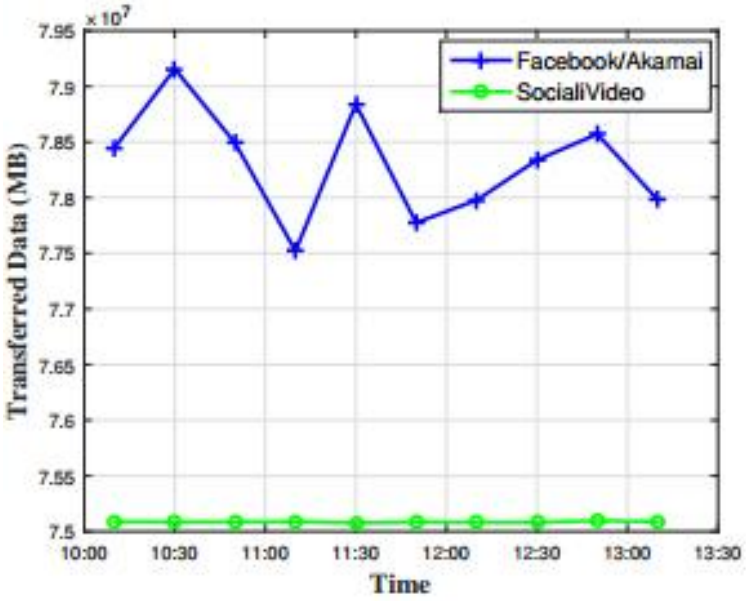


Figure 67 - Transferred traffic (#Bytes)

Another interesting point that we found in this experiment is that, average packet size of SiV is always larger than that of FB/AK approach. In the second experiment we look to the total transferred traffic in the delivery procedures to understand which approach is performing better. To this end, one HD video file of 70.13 MB is used and we collect the transferred data using Wireshark. Figure 67 illustrates the number of bytes (as the transferred traffic) which is transferred in the SiV and FB/AK approaches to deliver the 70.13 MB video file. The result shows FB/AK always uses more data traffic to transfer a video file, including the overhead generated due to low packet size than the SiV. On average, in-order to deliver the sample video file of the study, the FB/AK approach used 11.67% and the SiV approach used 7.01% more data to the actual size of the video. On average, 13,622 more packets were transferred in the FB/AK than in the SiV solution to deliver this video. Moreover, we found that on average, 4.4% of packets were re-transmitted in the FB/AK approach and in the SiV solution only

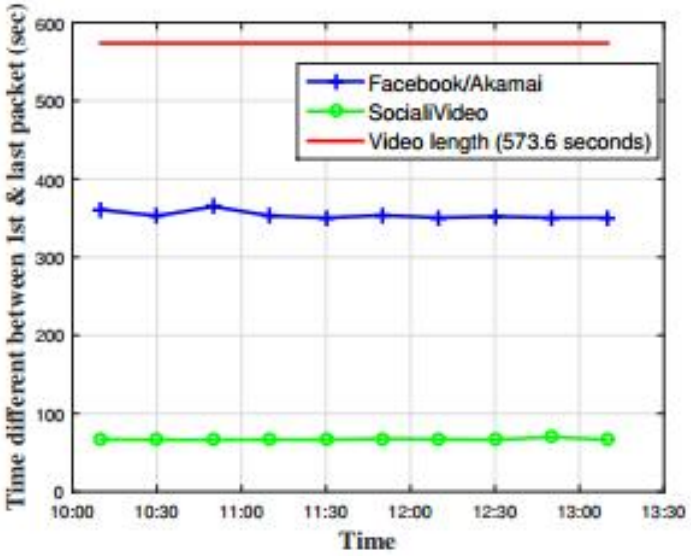


Figure 68 - Packet arrival time

0.12% re-transmissions were identified. Lastly, we analyzed total transmission time for both approaches using the previous sample HD video file. Figure 68 represents the time difference between first and last packet arrivals with respect to video length. Both approaches perform well in terms of the delay, but SiV transferred all the data packets in a short period of time in compare to FB/AK. Average packet transmission time for this simulation shows that SiV took only 67 sec to deliver the video whereas FB/AK used 354 sec. In summary, SiV shows better performance than the existing Facebook approach in terms of delay, total transferred traffic, and number of packets.

3.7.4 Conclusion

SocialiVideo is a novel solution to enhance the content delivery in CDN utilizing users' social information. The main objective of this approach is to enhance the multimedia communication by providing the possibility to stream a video directly from a user's premises in the case that two parties has a social connection and both are located the same location (e.g. network, city, country). We implemented a prototype of SocialiVideo based on Facebook content delivery and our performance evaluation shows the proposed approach reduces access delay and network load, performs better in terms of transmission time and provides a low cost and energy efficient solution for content providers and CDNs as well as better QoE for users. SocialiVideo can be merged as a complementary solution to the content delivery part of a large social networks such as Facebook and be combined with the existing CDNs and data centers to enhance their data delivery.

3.8 In-Network Caching in Content-Centric Networking (CCN)

3.8.1 Introduction

Recent years have seen, along with the growing popularity of video over Internet, a huge raise of traffic served by content-oriented networks. These network architectures such as CDNs and CCNs operate by replicating contents at several locations of the caches network, reducing the delay of delivery and improving the Quality of Service (QoS). CDN has become a basic layer in the network architecture through efficiently distributing content. Furthermore, CCN is a novel design for content-oriented networks by providing new protocols centered on the data. More material about CDN and CCN can be found in [90] and [91], respectively.
The introduction of in-network caches generates additional energy costs due to the mass memory access. Meanwhile they deliver important fraction of traffic, content-oriented networks need to be energy efficient. There have been several works to improve energy efficiency in such networks. Different models have been developed. In particular, it has been shown that important energy savings can be achieved by turning off network caches and links [92].
Here, we go on this idea by also considering performance evaluation of the content-oriented networks in arbitrary graph-based networks. A major step towards the success of this novel paradigm is to analyze and compare its performance with respect to the classical architecture of IP networks and its routing standards usually based on shortest path between the sources and the destinations. In this work, we give clear answers to this critical issue by proposing a methodology to assess how the innovative design of content oriented networks behaves as opposed to client-server architecture and its routing standards. At the same time, we study the impact of in-network cache parameters on energy consumption, on load, and on path length.
The problem we address in this work is called Energy Saving in Content-Oriented Networks (ESCON). It consists in finding the optimal subset of caches and links that could be turned off to minimize energy while finding a feasible routing in the network, which may not be necessary on a shortest path, under capacity constraints. Thus, we address the related object caching and traffic routing problem on arbitrary graph-based network topologies.

3.8.2 Optimization models and energy saving in content-oriented networks

Optimization models have been extensively used to study the performance of content-oriented networks. Particularly, they are extremely popular for addressing *the object placement problem* [92] [57] [93] [94] [10] [3] [11]. A description of these works is detailed in the sequel. We will also explain on which aspects our *object caching approach* differs from object placement problems.

Related Works: The Object Placement Problem

Here, the goal is to find the optimal locations of a set of objects to install in the network so that the total cost is minimal. Linear optimization technics are used in [57] to study the performance of a content distribution network modelled as a hierarchical cache system with a single origin server. A light-weight cooperative caches management algorithm was developed to maximize the traffic volume served from cache and to minimize the bandwidth cost. Mangili et al. [93] develop a novel optimization model to study the performance bounds of a content-centric network, by addressing the related object placement and routing problem. They show how the innovative design of a content-centric networking acts as opposed to the one proposed by content-distribution networks.
The authors in [92] propose to reduce energy consumption in CDNs by turning off CDN servers. It is shown that it is possible to reduce the energy consumption of a CDN while ensuring a high level of availability that meets customer demands. In order to study the impact of different memory technologies on energy consumption, the work in [11] focuses on the energy efficiency considering data delivery and storage. The authors propose a genetic algorithm approach for finding an energy-efficient cache location. As a consequence, CCN yields greater energy savings for very popular content. A further work in [10] discusses the energy benefit of using CCN by comparison to CDNs.
To optimize energy efficiency, operators try to switch off as many network devices as possible. Chiaraviglio et al. [94] consider the problem of minimizing power consumption for Internet Service Provider (ISP) networks, but they do not consider in-network caches. In particular, they propose and assess strategies to concentrate network traffic on a minimal subset of network resources. Given a

telecommunication infrastructure, the aim here is to turn off network nodes and links while still guaranteeing full connectivity and maximum link utilization constraints.
A more closely related work is the one from Araujo et al. [92], who propose a new model for saving energy by disabling equipment. The problem is to minimize the total energy consumption by turning off links and caches in order to find a feasible routing in the network satisfying all the demands under the capacity constraints. The authors study the impact of using in-network caches and content delivery network cooperation on an energy-efficient routing. Arbitrary network structures are considered taking into account in-router caches, while each cache serves only one city. Our problem differs from this optimization model in the sense that it enables sharing a single cache between multiple access points in a content-oriented network, but it also considers an "object caching approach".

Our Work: An Object Caching Approach

Our problem can be considered as two joint sub-problems: the *routing problem* and the *cache problem*. The routing problem aims at finding the links to be activated in order to satisfy all the demands with a minimum transmission cost. The cache problem is to find both the ON/OFF status of a cache and the volume that it intercepts for each demand, so that the storage cost is minimized. The objective of our work is to simultaneously optimize the traffic routing and the caching costs.
When inspecting the literature of the domain, we find that the related traffic routing and caching problem in their broadest sense have been widely studied. Nevertheless, in the literature, object placement problems are based on static caches which are characterized by a binary hit ratio indicating whether an object is placed or not. This is a major limitation, because here the caching is neither adaptive to the demand changes nor to the popularity of contents since the objects are statically placed.
Thus, this work is motivated by the fact that caching and routing should jointly adapt to traffic changes. In contrast to object placement which is static and based on placement decisions, our object caching approach is different. In fact, our caches are dynamic and self-adaptive to demand changes since they may run replacement algorithms such as Least Recently Used, Random Replacement, or Time-To-Live policies to add or remove contents from memories. Moreover, they are characterized by their average hit probabilities that indicate the fraction of demands served locally over all contents. Therefore, our work differs from object placement problems as treated in [92] [57] [93] [94] [10] [3] [11]. To the best of our knowledge, the related object caching and traffic routing problem on arbitrary graph-based network topologies has not been considered in the literature.

3.8.3 Problem Statement

We model the network by an undirected graph $G = (V, E)$, where V is the set of vertices and E the set of edges. A vertex can play the role of a provider, a user or a router. Let $P \in V$ be the set of providers, $U \in V$ the set of users, and $C \in V$ the set of routers.
We assume that each $p \in P$ has an unlimited capacity. The index r will refer to the caches, as we suppose that a cache is installed on each router $r \in C$. A cache $r \in C$ has a streaming capacity $b(r)$, and can be turned on or off. Turning on a cache gives rise to a fixed energy cost and an increased energy consumption in terms of load. Let β denote the power usage of a cache, which is the power consumption of a cache divided by the power consumption of a link [95]. We take $\beta \in [0,1]$. A cache is characterized by a *hit ratio*. As defined in [95], the hit ratio is the proportion of requests which are locally served by a cache with respect to the total number of requests. In our model, the hit ratio, denoted h_r, represents the maximal part of a demand served locally from a cache r. Each edge $uv \in E$ acts as a link and has a capacity $c(uv)$. We assume that each link uses one unit of energy.
We denote by K the set of demands. Each demand $k \in K$ is defined by a volume $D_k(gbps)$, a source $u \in U$, and a destination $p \in P$. Actually, k is a demand for several types of contents. In our work, we assume that the contents are aggregated and, then, we determine the part of D_k locally served by a cache r. This is done without making any decision neither on the type nor on the volume of the locally served content. Therefore, our hit ratio does not depend on the type of content requested. It is an indication on the maximal part of the demand served locally over all contents.
The goal is to find which caches and which links to turn off in the network to minimize energy consumption, in such a way that all the demands are satisfied respecting capacity constraints. The total energy cost, includes the energy used by the links and the energy used by the caches.

3.8.4 Mixed integer linear programming formulation

This section describes the optimization model we propose to save the total energy consumption, and simultaneously determine the optimal traffic routing and cache solutions. Let y_r be a binary variable which takes 1 if the cache in node $r \in C$ is turned on and 0 if not. Let x_{uv} be the binary variable which takes 1 if the link uv is activated and 0 if not. Let s_r^k, $r \in C$, $k \in K$, denote the proportion of the demand k cached by r. Also, let $z_r, r \in C$ indicate the load of the cache in equipment $r \in C$. By load we mean a fraction of used bandwidth. Finally, let f_{uv}^k, $uv \in E$, $k \in K$, be the flow of a demand k on the edge uv.
A portion of the total energy in the network is related to the transmission. The other part is related to the caches. Regarding the transmission costs, recall that each link uses one unit of energy. For the energy consumed by the caches, we assume that when a cache is turned on it uses a fraction Υ of β, and its power consumption grows linearly with load to reach β when fully utilized. The ESCON problem is then equivalent to the following MILP:

$$Min \ \sum_{uv \in E} x_{uv} + \sum_{r \in C} [\beta \Upsilon y_r + \beta(1-\Upsilon) z_r]$$

$$s_r^k \leq h_r D_k \qquad k \in K, r \in C \qquad (1)$$

$$\sum_{k \in K} s_r^k = b(r) z_r \qquad r \in C \qquad (2)$$

$$z_r \leq y_r \qquad r \in C \qquad (3)$$

$$\sum_{s \in N_r^k} f_{rs}^k - \sum_{t \in N_r^k} f_{tr}^k = \begin{cases} -D_k, \ if \ r = u \\ s_r^k, \ if \ r \neq \{u, p\} \\ D_k - \sum_{w \in C} s_w^k, \ if \ r = p \end{cases} \qquad k \in K, r \in V \qquad (4)$$

$$\sum_{k \in K} (f_{rs}^k + f_{sr}^k) \leq c(rs) x_{rs} \qquad rs \in E \qquad (5)$$

$$y_r \in \{0,1\} \qquad r \in C \qquad (6)$$

$$x_{uv} \in \{0,1\} \qquad uv \in E \qquad (7)$$

$$s_r^k \in \mathbb{R}_+ \qquad k \in K, r \in C \qquad (8)$$

$$z_r \in [0,1] \qquad r \in C \qquad (9)$$

$$f_{uv}^k \in \mathbb{R}_+ \qquad k \in K, uv \in E \qquad (10)$$

Constraints (1) express the fact that a cache $r \in C$ serves locally a part of any demand k up to its maximum hit ratio h_r. By constraints (2), the load of a cache is recorded. Constraints (3) indicate that the load cannot exceed the capacity and should be zero if the cache is off. Constraints (4) are the flow conservation constraints. Here, N_r^k is the set of neighbours of the cache r except users and providers that do not match the demand k. Finally, at the same time that they determine the ON/OFF status of links, constraints (5) indicate that the flow on a link cannot exceed the capacity.

3.8.5 Routing on shortest path based Heuristic

We propose a heuristic algorithm to solve our problem when considering a standard routing based on shortest paths between the source-destination pairs. The objective of this heuristic is to compare our MILP model solutions with the ones based on the classical shortest path routing. Our heuristic is a polynomial-time algorithm that for each demand computes a shortest path between its source and its destination. On this shortest path, the caching and the routing will be done. The algorithm is described in Figure 69. The comparison of this heuristic and our model is analysed later.

Data: Instance of the problem
Result: A problem solution $(s_r^k, z_r, y_r, f_{uv}^k, x_{uv})$
$s_r^k = 0, z_r = 0, y_r = 0, f_{uv}^k = 0, \quad x_{uv} = 0$
for $k \in K$ do
  $sp1 = shortest\_path(u_k, d_k)$;
  $already\_cached = 0$;
  for $r \in sp1 \cap C$ do
    if $\sum_{k \in K} s_r^k < b(r)$ and $already\_cached < D_k$
    then
      $s_r^k = \min(b(r) - \sum_{k \in K} s_r^k, D_k - \sum_{w<r} s_w^k, h_r D_k)$
      $already\_cached += s_r^k$;

```
        End
    End
End
```

Figure 69: Heuristic algorithm

3.8.6 Experimental Results

Instance Generation

First, we define the diameter of a network d as the length of the longest path between a user and a provider in the network. As a base for most of our instances, we consider a real French network of diameter $d = 4$ with $|V| = 66$ nodes and $|E| = 83$ edges. Table 18 shows the repartition of network nodes on the different hierarchical levels according to their role and number. For sensitivity analysis, we vary the parameters h_r and β of the cache. We assign to the caches of the same level the same hit ratio. We denote by h_1, h_2, h_3 the hit ratio assigned to caches of level $1, 2$ and 3, respectively. To be closer to the reality, we choose $h_1 > h_2 > h_3$. We define *the global hit ratio* $\widetilde{H}$ as the average of h_1, h_2, h_3. Finally, we consider 22 demands generated in such a way that in average populations among the cities behave similarly. Thus, the total amount of demands originating from a city is proportional to its population.

Table 18: Repartition of users, caches and providers on the network

nodes	number	level
Providers	0-3	4
Caches	5-7	3
	8-21	2
	22-43	1
Users	44-65	0

Preliminary Results

We implement our formulation on the ILP solver CPLEX version 12.5 [96]. Figure 70 shows the CPU time variation. As it appears the majority of our instances are solved in a short time. Only 8% of instances are solved in a time greater than 15 min; those cases arise only when $\beta = 0.3, 0.4, 0.5$.
For the instance with $h_1 = 0.4, h_2 = 0.3, h_3 = 0.2$ and $\beta = 0.3$, we look on how the streaming capacity of turned on caches is used by the different demands. According to Figure 71, $68.42\,\%$ of the on caches, are fully utilized. The memory not used doesn't exceed $43\,\%$ for partially utilized caches (32, 33, 36, 37, 41, 42). Therefore, our model guarantees a good use of the cache memory.

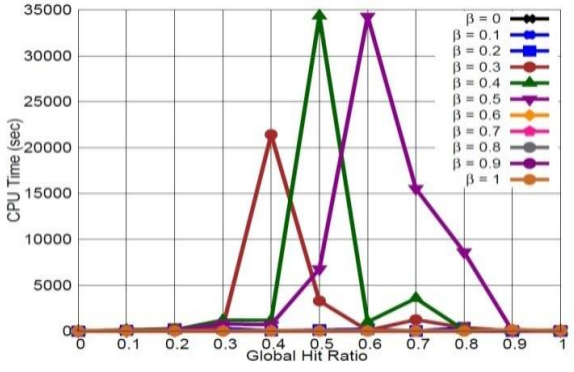


Figure 70: CPU time variation

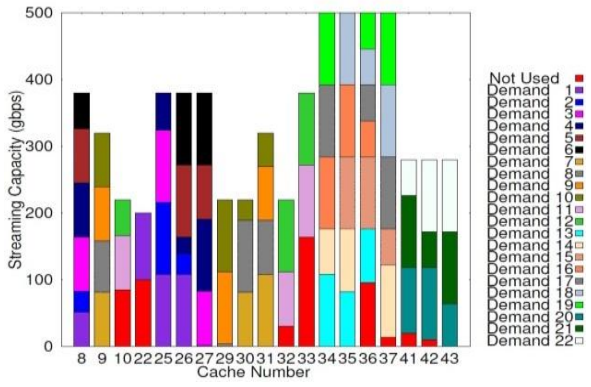


Figure 71: Distribution of the traffic load

We define $\mathrm{PG(k)}$ as the partial graph of a demand k; that is the partial graph obtained by keeping edges rs for which flow variables f_{rs}^{k} are nonzero. Experimentations show that the partial graph of each demand is a tree. Actually, the way in which we define our performance measures will depend on the nature of our solution. In fact, there are many different ways to measure performance of a network, depending on its topology. Our proposal is detailed in the next section.

Performance Evaluation

In order to evaluate the network performance, we consider four key metrics namely energy consumption, used bandwidth, effective hit ratio, and path length.

Energy Consumption

As previously stated, the energy consumption EC is computed as

$$EC = \sum_{uv \in E} x_{uv} + \sum_{r \in C} [\beta \Upsilon y_r + \beta(1 - \Upsilon) z_r]. \quad (11)$$

We exemplify the impact of the cache parameters. Figure 72 shows how energy consumption varies in terms of global hit ratio $\widetilde{H}$ and power usage β.
We obtain nearly the same energy consumption variation in terms of global hit ratio for all β. First, when $\widetilde{H}$ is zero, caches are not used. So the energy consumption is maximum. Second, when $\widetilde{H}$ increases between 0.1 and 0.8, EC decreases slightly. Finally, for caches with $\widetilde{H} = 0.9{,}1$, a fall of energy consumption is noted. Basically, a good cache usage allows reducing the energy consumption, whatever the values of the power usage β.

Bandwidth Utilization

Bandwidth is considered as the total data transfer rate, i.e. the amount of data that can be carried from one point to another in a given time period. In this usage, bandwidth refers to the data rate that is supported by the network connections. Total bandwidth in the network is denoted Bw and given by $Bw = \sum_{k \in K} \sum_{rs \in E} f_{rs}^{k}$. Figure 73 displays the variation of total bandwidth in terms of global hit ratio and power usage. We show the benefit of introducing caches in the network as a way to control congestion in the network.

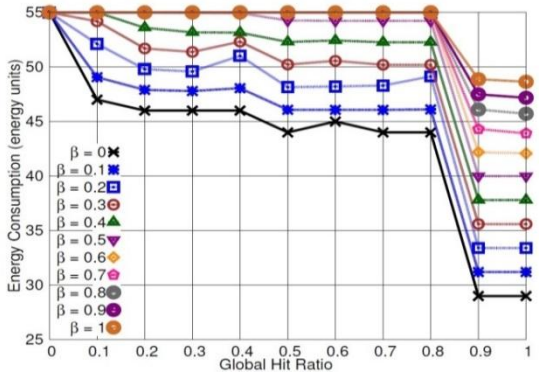


Figure 72: Impact of cache parameters on energy consumption

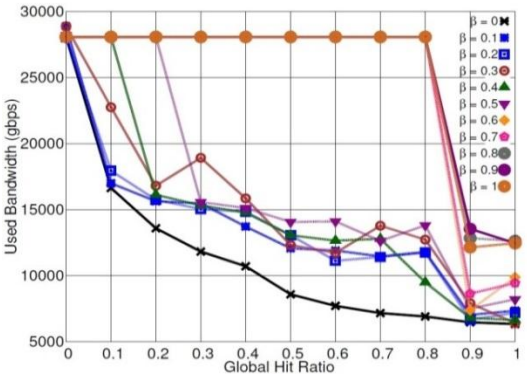


Figure 73: Impact of cache parameters on bandwidth utilization

Effective Hit Ratio

We propose to study the effective hit ratio for each demand. We distinguish the effective hit ratio and the real hit ratio. The real hit ratio, $\widetilde{h_r^k}$ is the probability of serving a demand k locally by a cache r. On the other hand, the effective hit ratio, h^{*k}_r is the probability of serving k locally by r knowing that it wasn't intercepted by its son in $\mathrm{PG(k)}$. Since $\mathrm{PG(k)}$ are trees, if $r \in \mathrm{PG(k)}$ there is a unique path between r and u. Let $\Pi_r = (r_n, r_{n-1}, \ldots, r_1, r_0)$ be this path, where $r_n = r$ and $r_0 = u$.

The effective hit ratio load as $h^{*k}_r = \begin{cases} 0, & r \notin \mathrm{PG(k)}, \\ \widetilde{h_r^k} \prod_{j=2}^{n} \left(1 - \widetilde{h_{r_{j-1}}^k}\right), & otherwise. \end{cases}$.

Here, $\widetilde{h_r^k}$ is given by $\widetilde{h_r^k} = \begin{cases} 0, & r \notin \mathrm{PG(k)}, \\ \frac{s_r^k}{\sum_{s \in A_r^k} f_{rs}^k}, & \text{otherwise.} \end{cases}$, A_r^k is the set of adjacent of r in $\mathrm{PG(k)}$.

Path Length

The Path Length refers to the number of intermediate devices through which data must pass between source and destination (number of hops). Calculating the path lengths depends on whether or not there exist caches in the network.

Let $k \in K$ and $\mathrm{PG(k)}$ its partial graph. The path length in the case of no caching is defined as the $\mathrm{PG(k)}$ tree height in terms of links. Indeed, when there are no caches in the network the demand crosses every level to be served by the provider. Now, denote L_i the different hierarchical levels in our real topology, $i = 1, \dots, d$. The path length with caches in the network load as

$$hop(k) = \sum_{i=1}^{d} \sum_{L_i \cap \mathrm{PG(k)}} (h^{*k}_r \prod_{j=1}^{n} \frac{f^k_{r_i r_{i-1}}}{\sum_{s \in A^k_{r_i}} f^k_{r_i s}}) \, i \, . \qquad (12)$$

Finally, we define an average path length for the whole network as $hop = \sum_{k \in K} \frac{D_k}{\sum_{k \in K} D_k} \, hop(k)$. Figure 74 shows a comparative study based on path length and average path length metrics. We investigate the cases with and without caches in content-oriented network. We observe that for $h_1 = 0.4, h_2 = 0.3, h_3 = 0.2$ and $\beta = 0.3$, introducing caches provides a gain of $56.11\,\%$ in terms of average distance. We can even reach $73\,\%$ of gain in term of distance with the demands $15,16,17$ and 18.

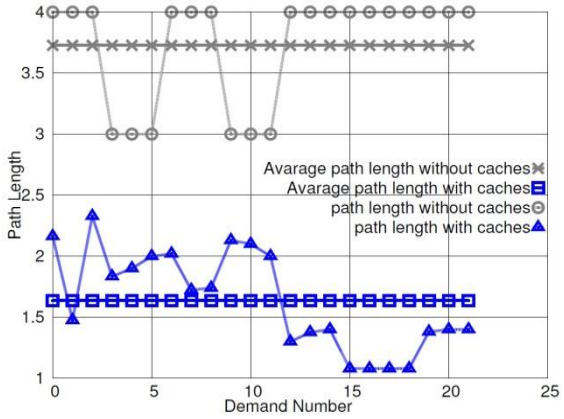


Figure 74: Path length and average path length of the instance $H = (0.4, 0.3, 0.2)$ and $\beta = 0.3$

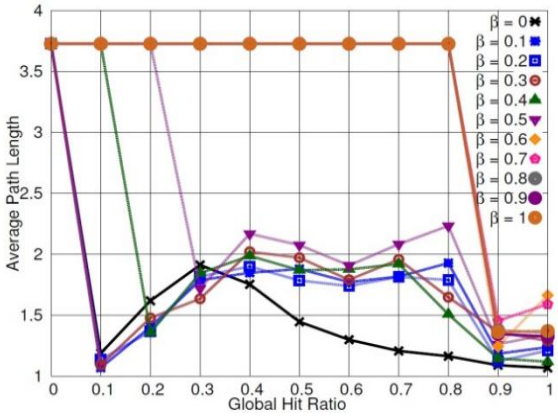


Figure 75: Impact of cache parameters on average path length

We exemplify now the impact of parameters of the cache. We look into how the average path length differs on changing values of the cache hit ratio h_r and of the cache power usage β. Figure 75 shows the gain in terms of distance provided by the introduction of caches in the network. Maximum gain of $76.49\,\%$ of the average path length, is obtained for $\widetilde{H} = 0.1$ and $\beta \leq 0.3$.

Our MILP model compared to a routing on shortest path-based caching model

Our heuristic allows us to compare some performance aspects with the MILP model. Here, we focus on the gain obtained in terms of energy consumption, cache usage and link usage by solving the model directly with CPLEX compared to shortest path-based solutions given by the heuristic. By cache and link usage we mean, the percentage of turned on caches and links in the network.

Figure 76 and Figure 77 display, respectively, cache usage and link usage for different cache parameters. We see that solving the model with CPLEX is clearly the best choice when the goal is to turn off as more possible the caches in the network. However, in terms of link usage the heuristic can give a better behaviour when $\beta = 0.7, 1$.

Now, we show that our heuristic still acceptable for some parameters even the performance issues. Figure 78 compares the values of objective function, while the computation time comparison is displayed in Figure 79. Here, the Gap means by what percentage the solution found by the heuristic is worse and how much time is saved by using it.

First, notice that when the network do not contain caches ($\widetilde{H} = 0$), it is feasible to solve the MILP optimally. For $\beta \leq 0.5$ we obtain solutions within at most $20\,\%$ of the optimum and save at least 68% of the CPU time. These solutions are more closely to the optimum (at most 5% of the optimum) when

the global hit ratio $\widetilde{H} \geq 0.7$. Now for $\beta \geq 0.6$ the heuristic gives solutions within at most $38.72\ \%$ of the optimum while saving more than 95% of the CPU time.
To conclude we can say that our model is the best choice when comparing it to the heuristic. Certainly, our heuristic is an acceptable choice for the network with $\beta \leq 0.5$ especially with a very short computation time. However, for $\beta \geq 0.6$ the shortest path-based solutions is not a good choice since we can even reach $38.72\ \%$ of energy saving by solving our model.

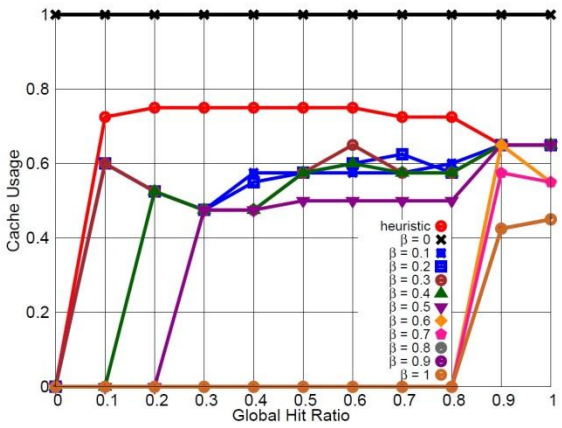


Figure 76: Comparison of cache usage given by the heuristic and the MILP model

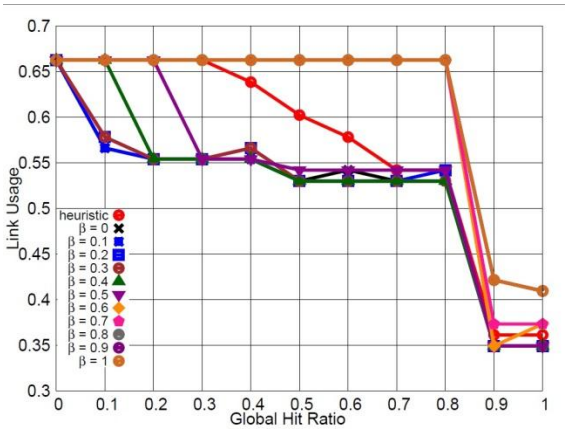


Figure 77: Comparison of link usage given by the heuristic and the MILP model

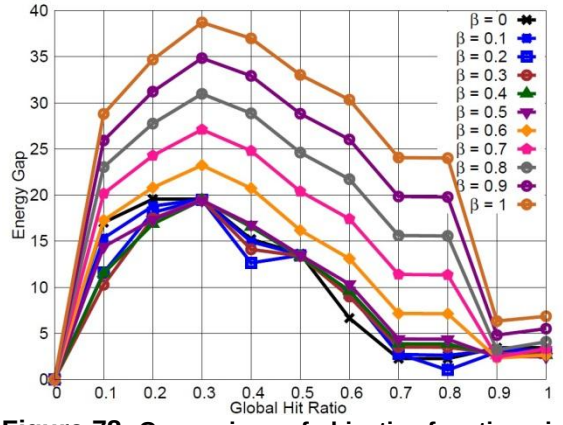


Figure 78: Comparison of objective function given by the heuristic and the MILP model

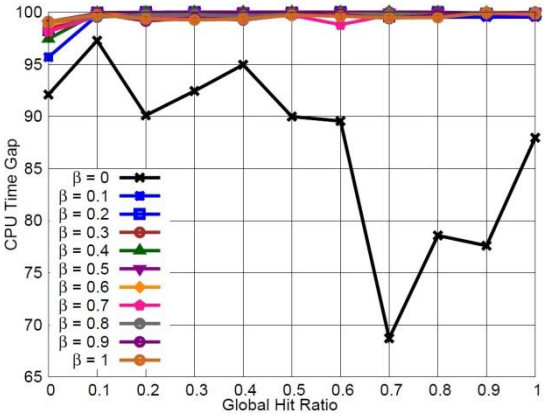


Figure 79: Comparison of CPU time given by the heuristic and the MILP model

3.8.7 Conclusion

We have proposed a new model for saving energy in content-oriented networks by disabling equipment. We have considered a caching object approach and jointly optimized the routing and caching object problem. Our model has been validated by solving instances based on real network topologies. Based on several network performance metrics, we have shown the impacts and the gains of introducing energy-aware on a real telecommunication network. Furthermore, we have proposed a heuristic allowing us to show the benefits of our model compared to the classical routing on shortest path-based caching model.

4 CLOUD NETWORKING

This section provides an overview of the state-of-the-art in energy-efficient cloud design (subsection 4.1). Then we propose a container-based and SDN controlled virtualization for cloud networking in subsection 4.2.

4.1 State-Of-The-Art

Santos et al [97] measure that a virtualized web server may consume 40 % more energy than its bare-metal (i.e. non-virtualized) counterparts running on a PC. It is worth noting that the context of the measurements are hypervisor based virtualization. The authors attribute majority of the overhead to networking, and show that they can reduce the energy overhead by over 16 % by buffering packets. The authors also measure Linux container-based virtualization using OpenVZ implementation which shows near bare metal performance. As containers can be more energy efficient than hypervised virtualization, we focus on container based virtualization in the remainder of this section.

Shea et al [98] compare Linux containers, as implemented by docker, with bare-metal performance using Rack PCs. The authors test different workloads: idle, Wordpress, PostgreSQL and Redis. When compared to bare metal, Docker increases average power consumption by two watts in the idle cases. However, considering the density plots of the publication, it appears that the power consumption with the other workloads has similar overhead (but increases variance). Compared to bare metal, docker adds 1 - 2 % (idle, Wordpress, PostgreSQL) or 10 % (Redis) to median energy consumption. So, interpreting the results, Docker adds a fixed overhead of a few percent to the workloads, with the exception of Redis with ten percent overhead, which is caused by dominating write() calls in the application, according to the authors.

Morabito et al [99] provide also some measurements using a PC, showing 1 watt overhead with docker when compared to bare metal. The authors benchmark Docker against hypervisor based virtualization, where Docker performs as well as hypervisors, but outperforms hypervisors in networking benchmarks. In another publication, Morabito et al [100] continue benchmarking of Docker with different workloads on system-on-a-chip computers, such as Raspberry Pi. In this work, the authors describe some upper bounds on maximal CPU loads that are feasible to run on a single system-on-a-chip computer after which new workloads should be assigned to other computers in the cluster.

4.2 Energy-Efficient Cloud Design

A current trend in implementing complex software is to split up the system into several independent components — micro-services. In video processing applications, the service typically takes the form of a chain, where the video stream is processed by a component at a time. This allows controlling for each component where the component runs. Because of the low overhead of containers, it is feasible to run a dedicated chain of containers for each stream. This allows controlling the processing of each stream with a fine granularity and to include only the processing components required for the particular stream. In our solution, we identify each device joining the network and set up a chain of containers for the particular device.

Our solution is based on an orchestrator and a SDN controller [101]. The orchestrator determines where the component should optimally be run. The orchestrator splits the workload between the network edge and network core and between the different machines in the data center. The SDN controller allows controlling the connections between the components. In particular, it allows the components to be relocated without interrupting the ongoing processing, and it allows new components to be inserted into the chain as needed. This allows an adaptive behaviour.

Some use cases are listed below:

- Inserting compression, rescaling or transcoding at the edge before the stream reaches a wireless link
- Inserting caching components at strategic locations
- Replacing transcoding components depending on the available bandwidth

- Moving components from one processing node to another processing nodes in order to switch off one of the nodes to save energy

We did a prototype implementation of this approach to study the performance and power consumption. The prototype processes a video stream and processing is split into a first phase done at an edge node and a second phase done in the core data center. The split point can be adjusted to control the fraction of the processing is done at the edge. The data center is the Ericsson Research data center in Lund, Sweden. The edge node is a Raspberry Pi 3 with 1 GB memory, a 4-core ARM Cortex-A53 CPU at 1.2 GHz. All components run Docker v1.12 containers. The video is MPEG-4 with a bitrate of 800 kb/s, 15 frames per seconds, and a size of 320x240 pixels without audio. The transport is RTP (Real-time Transport Protocol) over UDP. Each component decodes the MPEG-4 stream, processes the stream using a VLC based video filter, and encodes the MPEG-4 stream again. Filters include motion detection, color inversion, contrast and gamma adjustment, converting to greyscale. Each component uses the same format in this test to make the components interchangeable. We implement components are with an unmodified VLC to show that the platform works with standard dockerized software.

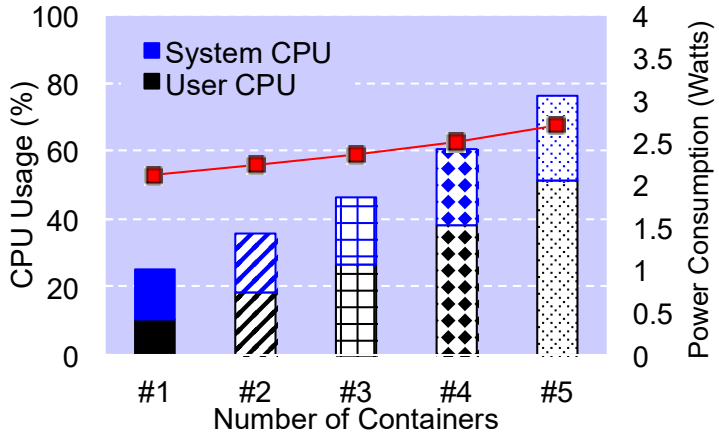


Figure 80: CPU usage and power consumption at an edge node under increasing number of video processing components.

The system is evaluated by moving an increasing number of components to the edge. Figure 80 presents the power consumption and the average CPU utilization of the edge node. We can see that the addition of components produces an almost linear increase of the CPU usage, each component adding 10% - 15% of CPU load. A relevant fraction of CPU load is due to operations in kernel space, mainly resulting from networking.

The idle power consumption of the Raspberry Pi 3 is about 1.3 W. This idle consumption can be seen as a reference value for comparing the overhead added by the platform and the components. The power consumption measured when only the platform is running is 2.12 W, with an increase of 0,82 W (63%) compared to the idle power consumption. For the first 3 components, the power consumption increases by 0.1 W per component, and the increase is slightly higher with the fourth (0.16 W) and fifth (0.2 W) component.
More interestingly, we notice that the increase per component is relative low compared to the load of the platform. This indicates that it is more power efficient to utilize as much capacity as possible of a node once it is switched on. This supports the idea of moving processing to a reduced number of nodes and switch off the unused nodes. Table 19 shows the total power when the same processing of 4 components is split between 1, 2, and 4 nodes. The “Total of active” column shows the total power when the remaining nodes are switched off. A power saving of up to 71 % can be achieved when the workload is concentrated to a low number of nodes.

Table 19: Power consumption when load is concentrated to a reduced number of nodes.

Nodes	Node 1, W	Node 2, W	Node 3, W	Node 4, W	Total, W	Total of active, W	Saving, W	Saving, %

4	2.22	2.22	2.22	2.22	8.88	8.88	0	0.00 %
2	2.32	2.32	2.12	2.12	8.88	4.64	4.24	47.75 %
1	2.58	2.12	2.12	2.12	8.94	2.58	6.36	71.14 %

5 SOFTWARE DEFINED NETWORKING (SDN) AND NETWORK FUNCTIONS VIRTUALIZATION (NFV)

This section presents the concepts, the state-of-the-art, and our contributions to the design of SDN/NFV. Our contributions are presented within three different directions each addressing a major challenge of SDN. These include: i) service functions chaining, ii) controller placement, and iii) load balancing.

5.1 State-Of-The-Art

In network architectures based on SDN the control plane is decoupled from the data plane while traditional network routers integrate both planes. The behaviour (control) of the network elements in the network is centralized in external devices called SDN controllers that communicate with the network devices through Application Programming Interfaces (API).
SDN brings flexibility and offers huge opportunities for network programmability thanks to its open interfaces. Thus, the migration of the networks towards SDN based networks is a key solution to manage and efficiently configure the network in order to cope with the huge increase of traffic and transport of new heterogeneous services with multiple Service Level Agreements (SLA). Furthermore, SDN could allow managing on/off functionality on routers interfaces that is a promising way to drastically reduce the energy consumption of the networks.

The three main challenges in the context of SDN are service functions chaining, controller placement, and load-balancing. In the following sections, we present detailed background and related work when describing our contributions to each challenge.

5.2 Energy Efficient SDN/NFV Design

This section presents our contributions to the design of SDN/NFV. Section 5.2.1 is focused on the service functions chaining problem and presents a novel mechanism to solve it through mixed integer linear programming. In Section 5.2.2, we investigate the so-called Controller Placement Problem (CPP) in the context of a Wide Area Network (WAN). The purpose is to find the best placement of controllers that allows managing a SDN based WAN. Section 5.2.3 is dedicated to the topic of geographical load balancing, and investigates the achievable gain of providing a sleep mode mechanism in DCs and the gain that can be achieved by using a geographical load balancing algorithm.

5.2.1 Service function chaining in SDN

Giving network-wide control over how switches handle traffic, SDN yields an elegant alternative that dramatically simplifies network management, providing abstraction of network devices and operations, thus more flexibility and dynamism in software-based network programmability. Handling a traffic not only means routing packet from a source to a destination, but also routing them through a specific ordered list of network services (service function chains) commonly called service functions (e.g. firewalls, proxies, DPIs...). Traditionally, service function chains are built by wiring cables between physical network functions (“middle-boxes"), often based on tightly coupled specific hardware and software. Network Function Virtualization (NFV) approach gives the alternative of implementing and deploying service functions as software that can run on generic hardware. Along with SDN, NFV provides solutions for routing packets through networks and install their corresponding service chains through their paths.

Nevertheless, they give rise to new challenges. High flexibility and programmability of networks through virtualization give means to provide network services while minimizing routing costs, memory of switches in terms of routing tables or function instantiations, while guarantying/ maximizing throughput and delay. A large research community both from Academia and Industry recently worked and are still presently working on these topics.
In this work, we focus on the service chaining problem and present a way of solving it through mixed integer linear programming. As the forwarding atomic rules implemented in [102] are independent and can be installed in any order along the chosen paths, we implement functions on paths while

respecting the order imposed by the service function chains each packet has to go through. In [103], candidate paths are first computed and then submitted to rule placement via decomposition and allocation on switches. In [102], a model in which candidate paths are provided (which is indeed a concrete case), and another where paths are computed with rule placement altogether are formulated. Concerning network service functions problems, in [104], virtual Deep Packet Inspection (DPI) engines are deployed as software by means of NFV while minimizing cost. In [105], solutions for outsourcing network service functions to a third-party cloud in order to reduce running costs on hardware, again via NFV, are investigated. In this work, we investigate the problem of the optimal deployment of service function chains in terms of maximal energy savings on a network by computing paths and function installation altogether.
A particularity of our abstraction is that we do not force packet routes to be unique: a route can split as many times as needed until it reaches destination, provided that the service function chain is correctly installed. We also do not accept packets to go multiple times through the same link. In the same way and for instance to avoid contingent forwarding rule table outrun, or delay limitation, our model also avoids creating cycles. We provide paths for each flow (commodity) and the corresponding service chain instantiation while minimizing costs in terms of power consumption. The instantiation of service functions (e.g. through virtual machines) on nodes has a certain cost, that is optimized by sharing functions between commodities and sharing instantiated nodes between functions. The use of wired cables for routing implies the intervention of routing tables upon which forwarding rules need to be implemented naturally using expensive and power-consuming Ternary Content Access Memory (see [103]). We include the consequent cost to our link utility. This is why our prior objective is to use the minimal number of links in the network. In [106], groups of commodities are solved in serial as partial problems in order to provide global solution. However, we try to identify classes of commodities that turn out “difficult" to route and chain and/or are most likely to show pacific coexistence with the upcoming commodities, and solve them in priority.

5.2.1.1 Problem statement

Preliminaries

We introduce first a slight description of the problem characteristics.

Network topology: The network is modelled as a bi-directed graph $G = (N, A)$, where each node $u \in N$ corresponds to a located switch provided or not with a device with a restricted memory capacity and each arc $a = (u, v)$ corresponds to a wired connection between two switches with a bandwidth limit.

Functions: A Service Function Chain consists of an ordered subset of the set of all service functions that could be required to deliver a service. In practice, one can meet no more than a dozen of those functions. Each service function chain can thus be seen as a precise protocol that a packet must go through during its route. For security questions or even high speed packet handling, we assume that a same instance of a service function cannot handle a number of packets beyond a certain limit. Thus each service function f is given with a number m_f representing the maximal number of packet type it can handle. Hence, if f is needed more than its capacity on one node, f needs to be implemented on that node several times. We denote F the set of functions that takes part into a global networking protocol.

Commodities: We consider a set of service sessions, each of which is associated with a certain service function chain that needs to be set between a source and a destination, along with a throughput threshold. We define a session k , or commodity, by a tuple (s_k, d_k, D_k, F_k) composed respectively of the source, the destination, the desired throughput and the service function chain associated to k.

Parameters and variables

- $G = (N, A)$ defines the network as a digraph. E defines the set of edges of G,
- $\forall u \in N$, $c_u \geq 0$ is the capacity of node u,
- $\forall (u, v) \in A$, $b_{uv} \geq 0$ is the bandwidth limitation of u to v arc,
- K is a set of sessions; $\forall k \in K$, $s_k \in N$ is its source, $d_k \in N$ is its destination, and D_k is a throughput threshold, measured by a required demand,

- F is a set of functions; $\forall k \in K$, $F_k \subset F$ is the required service chain to route k from its source to destination through the network,
- $\forall f \in F$, m_f is the maximal number of sessions f can handle on the same node,
- $\forall k \in K$, $\forall f \in F$, $p_{kf}$is an integer such that $\forall f, g \in F_k$, $p_{kf} < p_{kg}$ if f appears before g on the service chain of k,
- $\forall f \in F$, c_f is the memory space that f requires to be installed,
- $S \subset F \times F$, $(f, g) \in S$ if functions f and g are not allowed to handle the same packet on a unique node. This does not mean that f and g cannot be installed on the same node, but that they cannot be installed on the same node for the same session.

Let us define now the variables of the problem:

- $q_{uv}^k \in [0,1]$ is the fraction of D_k routed through arc (u, v)
- $Y_u^{kf} \in \{0,1\}$, that takes value 1 if the service function f has handled flow k during its route between s_k and u, 0 otherwise.
- $x_u^f \in \mathbb{N}$ the number of instantiations of f on node u.
- $y_u^{kf} \in \{0,1\}$, that takes value 1 if f handles flow k on node u, 0 otherwise.
- $\chi_{uv} \in \{0,1\}$, that takes value 1 if link (u, v) is used for traffic (in either way), 0 otherwise.
- $\chi_u \in \{0,1\}$, that takes value 1 if node u has, at least, one service function instantiation, 0 otherwise.

The service function chaining problem

The service chaining question can be formulated as follows: given a network topology, a set of functions, a set of commodities provided each with a service function chain, find the optimal global functions distribution over the network switches so that each commodity has a path through its service functions chain from source to destination, with correct throughput requirement. Global routing and function chain deployment needs to satisfy the arc's bandwidth limitation, and each switch memory limitations. We choose as objective to minimize the number of used link as primary objective and to minimize the number of used switches (a switch is used if at least one function is installed on it) as secondary objective.
This problem is illustrated for two commodities in Figure 81 and Figure 82, where the sources are represented as diamonds and the destinations as circles.

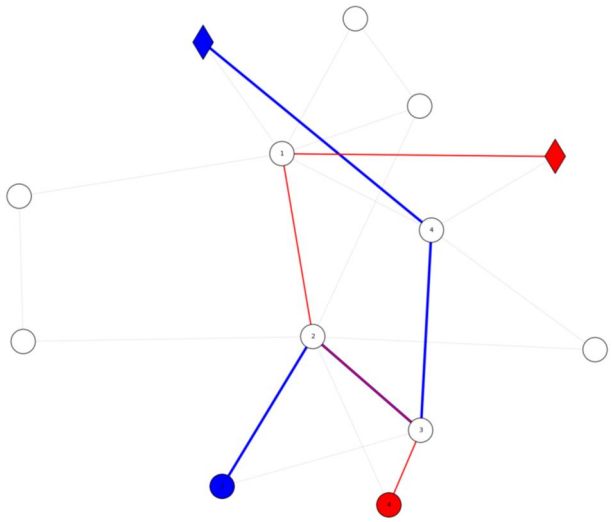


Figure 81: Route and chain solution example

In Figure 81, the route and chain solutions are given with bandwidth requirements and global limitation of 10, and node capacity 1, while each service function 1, 2, 3, 4 have size 1. The service Blue needs service chain 4321 while Red needs service chain 1234. They share functions 2 and 3 and have their own instantiation of functions 1 and 4.
In Figure 82, Red and Blue have the same service chain 1234, but each function has a capacity of 1. With link and ε-node (capacity 2) objective, each function has to be instantiated twice on each node.

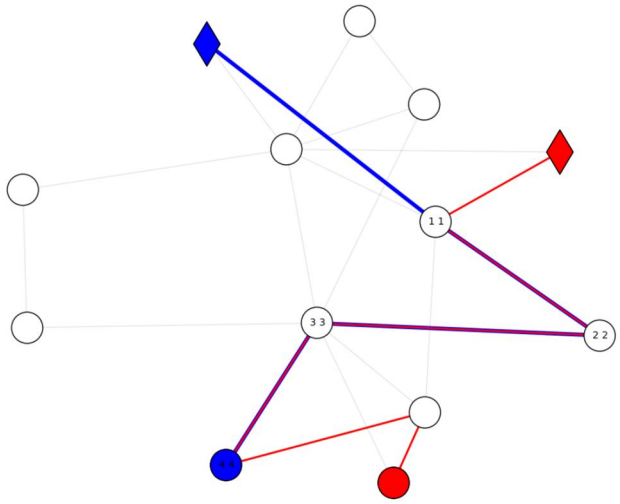


Figure 82: Route and chain solution example

5.2.1.2 MILP Formulation

The service chaining problem can thus be formulated as follows:

Minimize $$\sum_{(u,v)\in E} \chi_{uv} + \varepsilon \sum_{uEN} \chi_u \quad (1)$$

Subject to:

$$Y_{s_k}^{kf} = 0 \qquad \forall k \in C, f \in F_k \quad (2)$$

$$Y_{d_k}^{kf} = 1 \qquad \forall k \in C, f \in F_k \quad (3)$$

$$(q_{uv}^k - 1) + (Y_v^{kf} - Y_u^{kf}) \leq y_v^{kf} \qquad \forall k \in C, f \in F_k, \forall (\mathrm{u},\mathrm{v}) \in \mathrm{A} \quad (4)$$

$$(Y_u^{kf_1} - Y_u^{kf_2})(p_{kf_1} - p_{kf_2}) \leq 0 \qquad \forall k \in C, \mathrm{u} \in \mathrm{N}, f_1, f_2 \in F_k \quad (5)$$

$$y_u^{kf} + y_u^{kg} \leq 1 \qquad \forall k \in C, (\mathrm{f},\mathrm{g}) \in \mathrm{S} \cap F_k \mathrm{x} F_k, \mathrm{u} \in N \quad (6)$$

$$\sum_{f\in F} c_f x_u^f \leq c_u \chi_u \qquad \forall \mathrm{u} \in N \quad (7)$$

$$\sum_{k\in C} y_u^{kf} \leq m_f x_u^f \qquad \forall \mathrm{u} \in N, f \in F_k \quad (8)$$

$$\sum_{(u,v)\in A} q_{uv}^k - \sum_{(u,v)\in A} q_{vu}^k = \begin{cases} 1, & if\ u = s_k \\ -1, & if\ u = d_k \\ 0, & otherwise \end{cases} \qquad \forall k \in C, \mathrm{u} \in \mathrm{N} \quad (9)$$

$$\sum_{k\in C} q_{uv}^k D_k \leq b_{uv} \qquad \forall (\mathrm{u},\mathrm{v}) \in A \quad (10)$$

$$q_{uv}^k \leq \chi_{uv} \qquad \forall k \in C, \forall (\mathrm{u},\mathrm{v}) \in \mathrm{E} \quad (11)$$

$$q_{vu}^k \leq \chi_{uv} \qquad \forall k \in C, \forall (\mathrm{u},\mathrm{v}) \in \mathrm{E}$$

$$y_u^{kf}, Y_u^{kf}, \chi_{uv}, \chi_u \in \{0,1\} \qquad \forall k \in C, f \in F, u \in N, \forall (\mathrm{u},\mathrm{v}) \in \mathrm{A} \quad (12)$$

$$q_{uv}^k \in [0,1] \qquad \forall k \in C, \forall (\mathrm{u},\mathrm{v}) \in \mathrm{A} \quad (13)$$

$$x_u^f \in \mathbb{N} \qquad \forall \mathrm{f} \in F, u \in N$$

The objective, (1), consists in minimizing the number of active arcs and the number of active nodes. However, ε should be small enough such that the nodes cost does not compensate the use of additional arcs, as minimizing the route cost is our primary objective.

Constraints (2) and (3) state that at source, no commodity is yet handled by any Service Function Chain, and when destination is reached, all commodities went through their service function chain as required. The only way constraint (4) becomes non trivial is when $\left(Y_v^{kf} - Y_u^{kf}\right) = 1$. In that case, y_v^{kf} is automatically set to 1 if k is routed through (u, v).
Note that, according to constraint (26), at optimum, f is installed on the node v for commodity k if and only if $\left(Y_v^{kf} - Y_u^{kf}\right) = 1$ and $q_{uv}^k > 0$ for at least one adjacent node u of v. In particular, through a service chain, the profile of the variables Y^{kf} of a commodity along its route is (0 … 01 … 1). Then the function is installed on the first occurrence of 1. Constraint (4) also forces Y^{kf} to be non-decreasing along a path.
Constraint (5) uses the profile of Y variables to state that, through a service chain, the functions handle the commodities according to their relative position in the corresponding service chain: the first time Y^{kf_2} occurs with value 1, Y^{kf_1} is already non-zero, that is, the commodity k was already handled by f_1.
Constraint (6) states that two functions that cannot handle the same packet on the same node cannot be installed for this packet on the same node. While constraint (7) states that the capacity of nodes needs to be satisfied, constraint (8) forces f to be installed at least enough times to handle all commodities that need f on node u. Constraints (9) and (10) are the classic multi-commodity with bandwidth limitation constraints. Finally, constraints (11) are to turn on each arc (u, v) whenever a commodity is routed through it in either way.

5.2.1.3 The Path Finding and Function Placement Problems

Along with the original problem formulation, we present two important formulations to solve two alternative problems that are helpful variations of the original problem. The first alternative problem is the Path Finding Problem. It consists in finding routes through service chains that still need to be installed, but where a given configuration of functions installation is fixed. The aim is to route all commodities through their service chain from source to destination with bandwidth limitation constraints, only using the given configuration of function installation. Path Finding is thus an easier variation of the original problem. This model can be used in practice for a static configuration of routes when service function deployment is already configured and do not need to be altered. Path Finding can be formulated as:

$$Min \sum_{(u,v)\in E} \chi_{uv}$$

s.t (2)-(6), (9)-(11)

$$y_u^{kf}, Y_u^{kf}, \chi_{uv} \in \{0,1\} \qquad \forall k \in C, f \in F, u \in N, \forall (u,v) \in A$$

$$q_{uv}^k \in [0,1] \qquad \forall k \in C, \forall (u,v) \in A$$

We will refer to the second alternative problem as FunctionPlacement. Here, all the commodities are routed from source to destination with bandwidth limitation constraints, and the aim is to install, if ever feasible, on each path the corresponding service chain. The reason why we keep the data as a per-link rate for each commodity, namely q_{uv}^k, is because we consider that the routes can potentially split when, for instance, link capacities are too small or even when link utility objective suggests so. As the correspondence with path formulation of route from per-link rate to per-path rate is not one-to-one, we do not want to alter data and hence prefer to keep the per-link formulation. Objective is still node utility.
FunctionPlacement can hence be seen as a variant of the original problem, where routes are already given. In practice, it can be used to update service function configuration, for instance after adding several commodities dynamically to clean up space:

$$Min \sum_{u \in N} \chi_u$$

s.t (2)-(7)

$$y_u^{kf}, Y_u^{kf}, \chi_u \in \{0,1\} \qquad \forall k \in C, f \in F, u \in N$$

$$x_u^f \in \mathbb{N} \qquad \forall f \in F, u \in N$$

These formulations have been implemented and solved to optimality with the commercial solver cplex. We give some examples in the next section. The need for convergence acceleration is obvious, and we are investigating a couple of solutions for this particular problem by means of appropriate decompositions (Benders or Column Generation).

5.2.1.4 Experimental results

Machine characteristics:
We run all our instances on server (16 CPU Xem 2.13 Hz, RAM 32 Gb) and use ILOG CPLEX 12.3. We provide results with CPU time in seconds.

Instances:
Our main network topology is a graph with 33 nodes and 66 bidirectional edges. This graph has homogeneous arcs and nodes parameters. We deal with maximum 6-length service chains, that is 6 functions, all of size 1, with a separation condition for Function5 that can be possibly applied to the same commodity on same node only with Function6. All our bandwidth limitations are fixed at 10, and nodes will handle maximum 6 instantiations (their memory capacity is 6 and each function has size 1). These parameters correspond to realistic values.

We name our instances as netbBnN_KdD where B,N,K,D are respectively the bandwidth limitation, the maximal capacity of nodes, the number of commodities, the default throughput requirement (which will be the throughput requirement of every commodity). For instance, netb10n3_15d20 corresponds to bandwidth limitation 10 and node capacity 3 for 15 commodities each with demand 20. As a global demand of 10 for a global bandwidth of 10 is highly restrictive in terms of feasibility when the number of commodities grow, we did further experimentation with global demand of 2 in order to try and add as many as commodities as possible. We decided to associate with each commodity a maximal length service chain, that is, they all are of length 6, with a certain type. This choice leads to an extreme case which is of course harder to solve. Another complicating parameter is the capacity of each function to handle different packets at the same time. Fixing this data to a dozen seems realistic enough, yet we fixed it to 2 for Functions 1, 2, 3, and 4, to 3 for Function5 and to 5 for Function6. As a result, as the number of commodities grows, the minimal instantiation of service functions also grows quickly, which can lead to infeasibility in terms of node capacity. Hence, we built our instances upon extreme conditions to analyse our solutions.

Results and Discussion:
Table 20, Table 21 and Table 22 show the results. We ran all of our experiments with ε = 0.01. This gives a good idea of the solutions structure as the value's floor states the number of used arcs and its fractional part the number of active nodes.
Experiment 1 corresponds to our tests for 5, 6, 7, 8 and 9 commodities, with respectively node capacities 3 and 4, all with global 10 demands and global 10 bandwidth limit. Exact time and value are provided. In Experiment 2, we run instances with global demand of 5 or 15 for 2-9 commodities.
Next is our experimentation for adding commodities, with global demand of 2. As one can see, the minimal global memory allocation of nodes needs to be corrected quickly, partially because of the extreme choices that we made. The problem can't be solved exactly. Instead, we provide best integer solution found within 3 hours with the corresponding gap.
Table 20 shows the dramatic growth of the problem's complexity as node capacity becomes restrictive, even for small instances (less than 10 sessions). Table 21 points out performances

differences when instead; we stress the arcs capacities by forcing split, or relax them by fixing demands to 5. Table 22 clearly demonstrates the limitation of exact formulation for large instances. The gap for exact solving after three hours is very large, which motivate further investigation for acceleration procedure and development of heuristics.

Table 20: Results – Influence of nodes capacity

Instance	Time (s)	Value (nbArcs.nbNodes)
netb10n4_5d10	5	15.07
netb10n4_6d10	7	20.07
netb10n4_7d10	24	24.09
netb10n4_8d10	65	29.09
netb10n4_9d10	71	32.11
netb10n3_5d10	49	16.09
netb10n3_6d10	89	21.11
netb10n3_7d10	170	25.11
netb10n3_8d10	1267	30.12
netb10n3_9d10	213	33.14

Table 21: Results – Influence of links capacity

Instance	Time (s)	Value (nbArcs.nbNodes)
netb10n4_2d5	1	5.02
netb10n4_3d5	3	7.03
netb10n4_4d5	3	8.03
netb10n4_5d5	14	13.05
netb10n4_6d5	54	16.05
netb10n4_7d5	196	19.06
netb10n4_8d5	355	21.07
netb10n4_9d5	1275	23.07
netb10n4_2d15	14	11.04
netb10n4_3d15	16	14.06
netb10n4_4d15	55	17.08
netb10n4_5d15	416	29.08
netb10n4_6d15	62	35.10
netb10n4_7d15	731	42.11
netb10n4_8d15	None	None
netb10n4_9d15	None	None

Table 22: Results - Limitation of exact formulation

Instance	Time Limit (s)	Value (nbArcs.nbNodes)	Gap (%)
netb10n3_10d2	10 800	22.11	10.49
netb10n5_15d2	10 800	23.13	13.47
netb10n5_20d2	10 800	28.19	29.22
netb10n5_25d2	10 800	33.99	30.99
netb10n5_30d2	10 800	45.37	45.37
netb10n5_25d2	10 800	53.51	53.51
netb10n8_40d2	10 800	40.83	40.83

5.2.1.5 Conclusions and future work

We have developed a theoretical framework to address and solve the service chaining problem with minimal active resources in terms of network edges and nodes. The computation times to get the optimal solutions are prohibitive to solve real size instances, thus, we have proposed a heuristic algorithm based on PathFinding and FunctionPlacement formulations. Furthermore, we are

investigating exact procedure(s) such as Benders as it has shown effective in numerous MILPs. We can suggest that it will provide good performance for this problem.

5.2.2 SDN controller placement

The SDN controller of a network area is logically centralized but it must be physically distributed among several devices for scalability, performance and reliability reasons. Indeed, the communication between the farthest node and its controller must be within an admissible time response bound to meet QoS constraints. Furthermore, the capacity of a controller in terms of processing, memory and in/out bandwidth limits the number of nodes (e.g. switch, routers) that could be managed by a single controller. Then, SDN deployment, particularly in a Wide Area Network (WAN), raises some open and challenging questions such as: how many controllers are required to manage the whole network? What are the clusters of nodes depending on each controller? What are the right nodes to place them in the cluster? And so on.

In this work we investigate the so-called Controller Placement Problem (CPP) in the context of a wide area network. The purpose is to find the best placement of controllers that allows managing a SDN based WAN. This problem has been extensively studied in the last 3 years. Several variants have been proposed depending on different optimization criteria and additional constraints: minimum number of controllers, minimization of the worst case latency between nodes and controllers, minimization of the inter-controller latency, optimization of the balancing of clusters of routers for each controller, taking into account or not the controllers capacities, and finally considering failure of both controllers and routers.
In its simpler form the Controller Placement Problem reduces to a Facility Location Problem and is proved to be NP-Hard [107]. This study presents innovative formulations for the optimization of the placement of controllers in a resilient SDN architecture. Several optimization criteria are embedded in the formulations (in the objective function or as a constraint). Our main objective is to minimize the number of active controllers needed in a WAN while considering several levels of back up controllers and maintaining tight latency, capacity and balancing constraints. Our formulations should be used to help network architects to find the best deployment of controllers in realistic networks. Our main objective is to minimize the number of active resources (energy) that translates in minimizing the number of controllers which has an increased impact in case of over provisioning of SDN controllers for resiliency purpose.

5.2.2.1 Problem Statement

Performance criteria
The main idea driving the development of SDN is to separate the data plane from the control plane and thus to provide an abstraction between the controller (control plane) and the network elements. This controller acts as the "brain" of the network: it centralizes the control plane policies, and instantiates the routing rules for each network elements through an API. This network architecture brings huge flexibility. However it induces some side effects regarding the latency between the controller and its network elements and the reliability of the network. Indeed, if the controller fails, the whole network could be down. To prevent such a drawback it is necessary to use redundant controllers, leading however to solutions with multiple under-used controllers. Our purpose in this work is to find the best trade-off between various optimization criteria thanks to the use of integer linear programming.
Several metrics can be used to provide a good controller placement on a network, depending on the considered use case: for instance in a data center the latency is not a tight constraint while the reliability is fundamental.

The network is modelled by an undirected graph $G = (V, E)$ where $V = \{1, \dots, n\}$ is the set of nodes (routers) and E is the set of edges. The edge uv corresponds to the bidirectional link between nodes u and v and its weight, denoted d_{uv}, represents the latency of the link.
Assuming that a controller could be placed at any node, the length of the shortest path from a router r to its controller c corresponds to the propagation latency and is denoted by $d(r, c)$. Assuming k controllers that can be dispatched throughout the network at any of the possible locations $C = (C_1, C_2, \dots)$, some usual metrics are defined by:

- The average latency between a node and its controller:

$$L_{avg}(C_i) = \frac{1}{n} \sum_{v \in V} min_{c \in C_i} d(v,c) \quad (1)$$

- The maximal or worst-case latency between a node and its controller:

$$L_{max}(C_i) = max_{v \in V} min_{c \in C_i} d(v,c) \quad (2)$$

- The maximal latency among all the couples of controllers:

$$L_{cc}(C_i) = max_{C_1, C_2 \in C_i} d(c_1, c_2) \quad (3)$$

- The load balancing of the controllers:

$$EC(C_i) = max_{c \in C_i} n_c - min_{c \in C_i} n_c \quad (4)$$

Where n_c is the number of nodes managed by controller c.

A good controller's placement solution is a placement in which one or several of these criteria are minimized.

Related Work

In 2012, Heller et al. addressed first the Controller Placement Problem in WAN [107], taking into account the control plane propagation latency. They study three placement metrics: the minimum average latency (1) is obtained by solving a minimum k-median problem, the worst case latency (2) by solving the related minimum k-center problem and finally the number of nodes within a latency bound, by solving then a maximum k set covering problem. This stuy motivates the relevance of the controller's placement problem and quantifies the impact of this placement on real topologies with up to 50 nodes.
However some structuring metrics like inter-controllers latencies (3) or the load balancing (4) are not taken into account.
Several works by Hock et al. [108] and Lange et al. ([109] and [110]) deal with a multi-objective approach. They include the inter-controllers latency (3), load balancing (4) and resilience among the decision criteria. They developed a MATLAB framework named POCO for Pareto-based Optimal COntroller Placement. This framework allows displaying the Pareto frontier enumerating all the feasible placements and visualizing the different Pareto optimal placements. They also developed heuristics to deal with larger networks. But their purpose is to find the best placement of a set of fixed controllers, not to minimize their quantity.
Yao et al. [111] consider a capacity on the load of controllers and proposed a capacitated K-center algorithm, taking into account the average latency and the controller capacity as a constraint.
In [112], Cervello-Pastor et al. propose an algorithm to find the minimum number of controllers within a delay limit in the worst case scenario of a single controller failure. Ros and Ruiz, in [113], consider a failure probability of each component and propose a heuristic to determine the best placements to achieve five nine reliability between controllers and nodes. Their study shows that the reliability is strongly dependent to the network topology structure, particularly the graph density.
More generally, the controller placement problem is a variant of the well-known facility location problem [114]. As explained in [107], minimizing (1) and (2) corresponds respectively to the k-median problem and to the k-center problem. While minimizing the number of controllers refers to the set-covering problem. Several studies [115], [116], [117]) introduced failure in the delivery network, referring then to the fault-tolerant facility location Problem.

The increasing amount of the literature on the Controller Placement Problem in the last 3 years shows the growing interest of the SDN community for providing a good controllers placement. For small instances it is possible to enumerate all feasible solutions. In this case, one can derive the entire Pareto front in order to consider the whole criteria. However some previous works have shown that these objectives often give opposite solutions in terms of placement. Furthermore, enumerating all feasible solutions is obviously not realistic when working on a real network.

Our objective in this work is to use linear programming tools to provide both the optimal number of controllers and their optimal placement within the network. Our approach allows considering several criteria.

5.2.2.2 The Controller Placement Problem

The minimization of the number of active controllers requires shifting in the constraints the network performance criteria so that the QoS requirements are still guaranteed. Thus, we assume that the routers are automatically assigned to their nearest available controller and a maximal latency constraint is introduced to ensure good performance. In a pre-processing procedure, the entire candidate nodes from which a router could be reached within the maximal required latency is computed. The formulation aims, then, at finding the minimal number of active controllers such that each router node is assigned to one of its closest controller, and such that all the controllers have an equivalent number of nodes to manage (the so-called load balancing constraints).

Integer linear Programming Formulation

Let C be the set of candidate nodes for placing the controllers indexed by i and R the set of routers nodes j (C is a subset of R). D is the latency matrix: $d_{(i,j)}$ being the length of the shortest path between i and j. The maximal allowed latency between a router and its controller is denoted l_{max} and the maximal latency between two controllers is noted l_{cc-max}. The maximal difference between the loads of two controllers is δ. The cover matrix A of graph G is defined as:

$$a_{ij} = \begin{cases} 1 & \text{iff } d_{ij} \leq l_{max} \\ 0 & otherwise \end{cases}$$

To ensure that the router j will be assigned to one of its nearest controller, we sort all the candidate sites i by increasing order of d_{ij} and we note σ_{jq} the index of candidate site in position q.
The shortest paths matrix, D, can be computed with Djikstra algorithm [118] for each couple of nodes (complexity $\mathcal{O}(mn + n^2 \log(n))$ or using Floyd-Warshall algorithm [119](complexity $\mathcal{O}(n^3)$).

The three binary decision variables $\forall i, i' \in C, \forall j \in R$ are:

- $x_{ij} \in \{0,1\}$, that takes value 1 if controller i is assigned to router j, 0 otherwise.
- $y_i \in \{0,1\}$, that takes value 1 if node i is active (it has a controller), 0 otherwise.
- $t_{ii'} \in \{0,1\}$, that takes value 1 if $y_i = y_{i'} = 1$ ie if both nodes i and i' are active, 0 otherwise.

A formulation for the Controller Placement Problem embedding all the metrics as side constraints is:

$$\min \sum_{i \in C} y_i$$

st.

$$\sum_{i \in C} a_{ij} y_i \geq 1 \qquad \forall j \in R \qquad (5)$$
$$\sum_{i \in C} x_{ij} = 1 \qquad \forall j \in R \qquad (6)$$
$$x_{ij} \leq y_i \qquad \forall i \in C, \forall j \in R \qquad (7)$$
$$y_{\sigma_{jq}} \leq \sum_{m=1}^{q} x_{j\sigma_{jm}} \qquad \forall j \in R, \ \forall q \in [1, |C| - 1] \qquad (8)$$
$$t_{ii'} d_{ii'} \leq l_{cc-max} \qquad \forall i, i' \in C \qquad (9)$$
$$-\delta - |R|(1 - t_{ii'}) \leq \sum_{j \in R} \left(x_{ij} - x_{i'j}\right) \qquad \forall i, i' \in C \qquad (10)$$
$$\sum_{j \in R} \left(x_{ij} - x_{i'j}\right) \leq \delta + |R|(1 - t_{ii'}) \qquad \forall i, i' \in C \qquad (11)$$
$$t_{ii'} \geq y_i + y_{i'} - 1 \qquad \forall i, i' \in C \qquad (12)$$
$$t_{ii'} \leq y_i \qquad \forall i, i' \in C \qquad (13)$$
$$t_{ii'} \leq y_{i'} \qquad \forall i, i' \in C \qquad (14)$$
$$x_{ij}, y_i, z_j, t_{ii'} \in \{0,1\} \qquad \forall i, i' \in C, \ \forall j \in R \qquad (15)$$

Constraints (5) ensure that each router must be covered by at least one controller within the latency bound. Constraints (6) and (7) assign each router to exactly one controller, while constraints (8) ensure that the routers are assigned to their nearest available controller. (9) is the constraint on the maximal allowed inter-controller latency. (10) and (11) are the load balancing constraints in terms of number of assigned routers ie the load difference between two controllers can't be more than a fixed threshold δ. (12), (13) and (14) are used to define the variables $t_{ii'}$ that are necessary to make the formulation linear ($t_{ii'} = y_i\, y_{i'}$ is required to measure the difference of load between each couple of clusters of routers). Finally (15) defines the binary variables.

Numerical results

This formulation has been run on instances issued from SND Lib [120], using this formulation and the commercial solver CPLEX [96]. Below some results for the instance Cost which is composed of 37 nodes and 57 edges are illustrated. The optimal solution, obtained with parameters l_{max} =30% and l_{cc-max}=70% of the graph diameter, and δ=2, appears in the Figure 83.
Figure 84 illustrates the fact that relaxing the constraint on the load balancing of the controllers clusters doesn't allow to save controller (still 4 optimal controllers) while it yields to very unbalanced clusters, from 3 to 18 nodes.

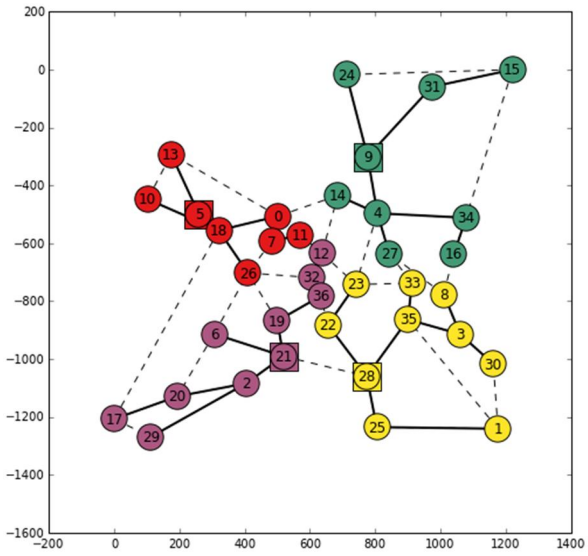


Figure 83: Optimal solution with lmax=30%, lcc-max = 70% and δ =2

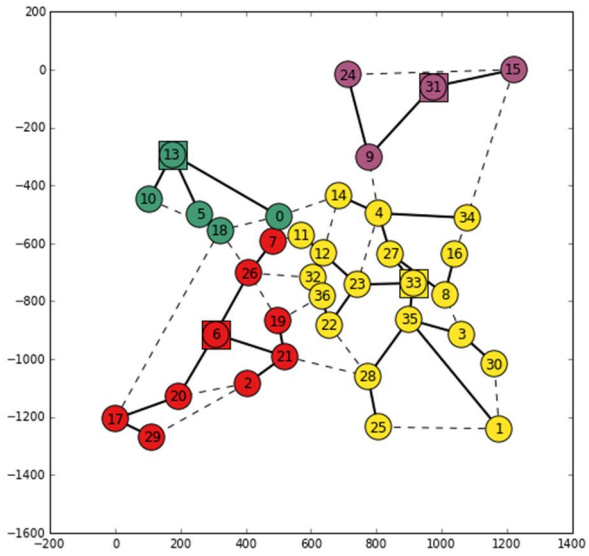


Figure 84: Optimal solution with lmax=30%, lcc-max = 70% and δ =37

Table 23 shows the sensitivity analysis of the three main parameters of the problem. Obviously the number of controllers increases when the maximal latency l_{max} decreases: from 2 controllers when l_{max}= 0.4 to 8 controllers when l_{max} = 0.25. The latency between each couple of controllers l_{cc-max} is more restrictive to obtain a feasible solution and when its value is increased it changes the localization of the controllers, particularly it yields to solutions where controllers are close to the graph border.

Table 23: Sensitivity analysis

l_{max}	l_{cc-max}	δ	Placement
0.4	0.7	3	4 21
0.3	0.7	3	9 18 21 28
0.27	0.7	3	2 4 11 26 28 34
0.25	0.7	3	2 9 18 22 23 24 26 30
0.35	0.4	3	∅
0.35	0.7	3	10 19 22 29

0.35	0.9	3	14 16 19 32
0.35	1.0	3	2 5 8 9
0.35	0.7	1	2 3 5 9
0.35	0.7	7	6 9 28
0.35	0.7	15	6 31 35

We used the framework POCO [108] to evaluate the solutions on the three main performance criteria. These evaluations are illustrated in Figure 85 and Figure 86, extracted from the framework. The green points correspond to the optimal placement with routers assignment to their nearest controller. The red points correspond to the same placement but taking into account the additional constraints (load balancing, latency). It shows that the solutions given by the linear programming formulation give a very good trade-off between the maximal latency (2), the balancing of clusters (4) and the inter-controllers latency (3). The green coloured solution is near the pareto-frontier for the 3 metrics. Nevertheless, the assignment to the nearest controller could increase the maximal latency if it allows minimizing the average latency. The red coloured solution is out of the pareto front in Figure 85 as routers are assigned to their second controller to satisfy load balancing constraints.

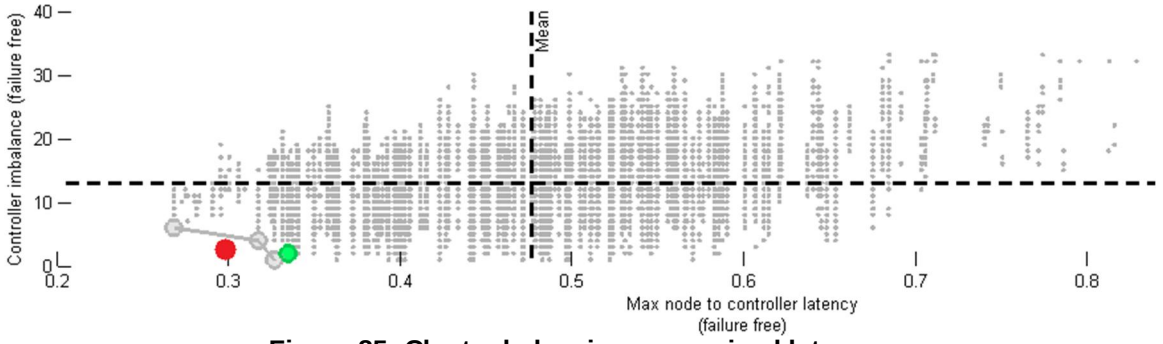


Figure 85: Cluster balancing on maximal latency

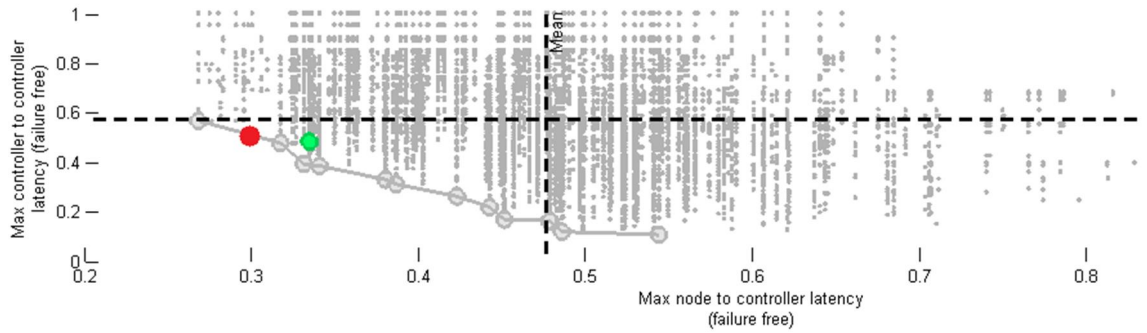


Figure 86: Inter-controller latency on maximal latency

The gain obtained in relaxing the latency constraints is illustrated in the following figures Figure 87 and Figure 88, which represent the placement of controllers in the instance ZIB from SND Lib of 54 nodes. All instances from SND lib were solved to optimality in a few seconds.
This formulation is extended to the failure case in the next section.

5.2.2.3 The Resilient Controller Placement Problem

If a controller fails, its routers must be assigned to another one, leading then to an increase of the latency between routers and controller and potentially to unbalanced domains, especially if the secondary controller takes the management of all the routers of the failed controller.
In this section we consider a failure probability for each controller assuming that these failure probabilities are inherent to the controller and independent from each other.

Problem Formulation

Let p be the failure probability of the controller.
We define the new following variables:

- $x_{ij}^k \in \{0,1\}$, that takes value 1 if controller i is the k^{th} backup controller of router j, 0 otherwise.
- $z_j^k \in \{0,1\}$, that takes value 1 if j has a $(k-1)^{th}$ backup controller but not a k^{th} backup controller, 0 otherwise.

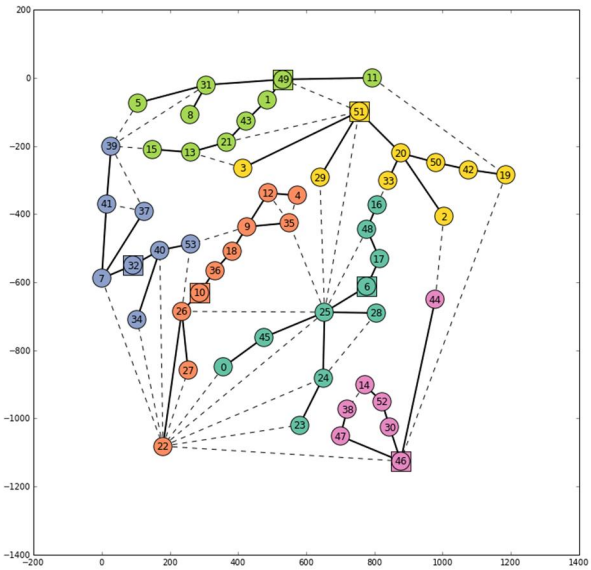


Figure 87: Optimal solution with $l_{max} = 30\%$, l_{cc-max} =70% and δ=3

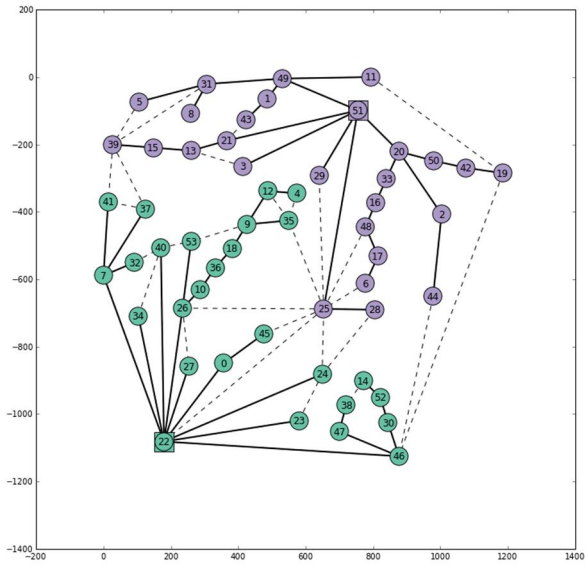


Figure 88: Optimal solution with $l_{max} = 50\%$, l_{cc-max} =70% and δ=3

The router j is assigned to its k^{th} controller if and only if all the $(k-1)^{th}$ primary controllers fail.
The probability that all the controllers of a router j fail until level $k-1$ is p^{k-1}. In that case, if j is assigned to controller i at level k then it is operational with probability $1-p$. Then the delay of communication between the k^{th} controller i and j is

$$\sum_{i \in C} d_{ij} x_{ij}^k (1-p) p^{k-1}$$

When there is no controller at level k for router j it induces a penalty cost. It is given by:

$$l_{max}\, z_j^k p^{k-1}$$

Finally, our main objective is still to minimize the number of active controllers, to which the delays are added. Thus, a formulation for the resilient controller placement problem is:

$$\min \sum_{i \in C} y_i + \sum_{j \in R} \sum_{k=1}^{|C|} \sum_{i \in C} d_{ij} x_{ij}^k (1-p) p^{k-1} + l_{max} \sum_{j \in R} \sum_{k=1}^{|C|+1} p^{k-1} z_j^k$$

Subject to:

$$\sum_{i \in C} x_{ij}^k + \sum_{s=1}^{k} z_j^s = 1 \qquad \forall j \in R, \forall k \in [1, |C|+1] \quad (16)$$

$$\sum_{i \in C} x_{ij}^k = 1 \qquad \forall j \in R, \forall k \in [1, \gamma] \quad (17)$$

$$\sum_{k=1}^{|C|} x_{ij}^k \leq y_i \qquad \forall i \in C, \forall j \in R \quad (18)$$

$$-\delta - |R|(1 - t_{ii'}) \leq \sum_{j \in R} \left(x_{ij}^k - x_{i'j}^{k'} \right) \qquad \forall i, i' \in C, \forall k, k' \in [1, \eta] \quad (19)$$

$$\sum_{j \in R} \left(x_{ij}^k - x_{i'j}^{k'} \right) \leq \delta + |R|(1 - t_{ii'}) \qquad \forall i, i' \in C, \forall k, k' \in [1, \eta] \quad (20)$$

$$d_{ii'}\, t_{ii'} \leq l_{cc-max} \qquad \forall i, i' \in C \quad (21)$$

$$t_{ii'} \geq y_i + y_{i'} - 1 \qquad \forall i, i' \in C \quad (22)$$

$$t_{ii'} \leq y_i \qquad \forall i, i' \in C \qquad (23)$$
$$t_{ii'} \leq y_{i'} \qquad \forall i, i' \in C \qquad (24)$$
$$x_{ij}^k, y_i, z_j^k, t_{ii'} \in \{0,1\} \qquad \forall i, i' \in C, \forall j \in R, \forall k \in [1, |C| + 1] \qquad (25)$$

Constraints (16) ensure that a router j is either assigned to a controller or receive a penalty at each level k. (17) guarantees that each router has a minimum of γ levels of back up controllers. (18) prevents a router to be assigned to the same controller at two different levels.
(19) and (20) are the load balancing constraints from level 1 to level η. It means that until level η each time a controller fails its routers are re assigned to several controllers such that difference in number of routers of two different controllers doesn't exceed δ. (21) is the constraints on the maximal latency between two controllers and finally, (22) and (23) are required to define the variables $t_{ii'}$.

The objective function can be re written as

$$\min (1 - \varepsilon)\, N(y_i) + \varepsilon\ L(x_{ij}^k, z_j^k)$$

where $\varepsilon \in [0,1]$.

This is a bi-objective problem that is solved in two stages:

- In the first step the problem restricted to our first objective $\min \sum_{i \in C} y_i$ is solved with an additional constraint on the maximal latency between a router and its controller. It provides the optimal number of controllers N^*.
- In the second step the problem with our secondary objective

$$\sum_{j \in R} \sum_{k=1}^{C} \sum_{i \in C} d_{ij} x_{ij}^k\ (1-p)p^{k-1} + l_{max}\ z_j^k p^{k-1} \qquad (26)$$

  is solved and instead of the constraint on the delay (which is embedded in the objective function) a constraint to ensure that the number of controllers must be less or equal than the optimal one N^* is added.

Numerical results

The formulation has been tested on the SNDlib instances and, for sensitivity analysis, on around 3.000 simulations on random graphs. The parameters used to run the simulations are given in Table 24. In order to test very resilient architecture the failure probability has been fixed to $p = 0.95$. Indeed, when p is around zero the formulation tends to give very few back up levels.

Table 24: Parameters used for simulations

Parameters	Value
Number of nodes	[10, 80]
l_{max}	{3 000, 5 000, 7 000}
l_{cc-max}	7 000
δ	3
γ	2
η	1

The number of controllers on the number of nodes is illustrated in Figure 89 and the number of controllers on the number of arcs in Figure 90. It makes clearly appear that the diameter of graph increases with the number of nodes, resulting in the need for more controllers. At the opposite when graphs are denser (ie more arcs) there are more possible connections between routers and controllers leading then to fewer required controllers.

In Figure 91 the optimal number of controllers is represented depending on the length of the average shortest path. Each of the three colors corresponds to a set of solutions obtained with a value

parameter of l_{max} (3.000 in blue, 5.000 in green and 7000 in red). For high values of l_{max} most of the solutions required few controllers. Furthermore, the number of required controllers increases with the average length, until no more solutions are found.
When the maximal latency is 3.000 up to 14 controllers could be required; while for high maximal latency the number of controllers decreases to the minimum threshold 2 (one back up controller).

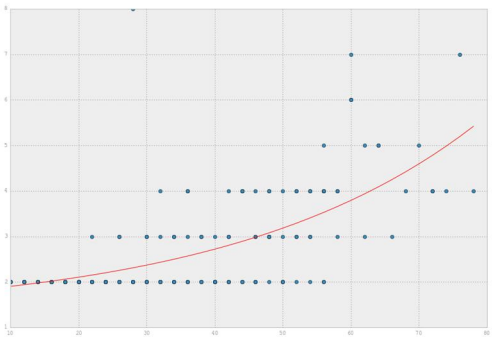


Figure 89: Number of controllers on the number of nodes

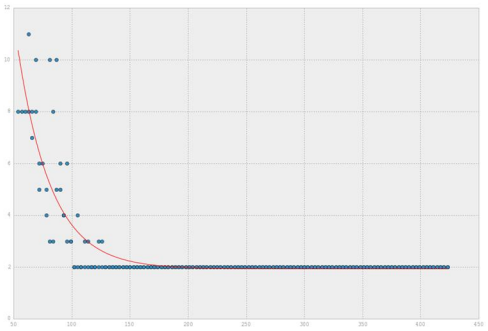


Figure 90: Number of controllers on the number of arcs

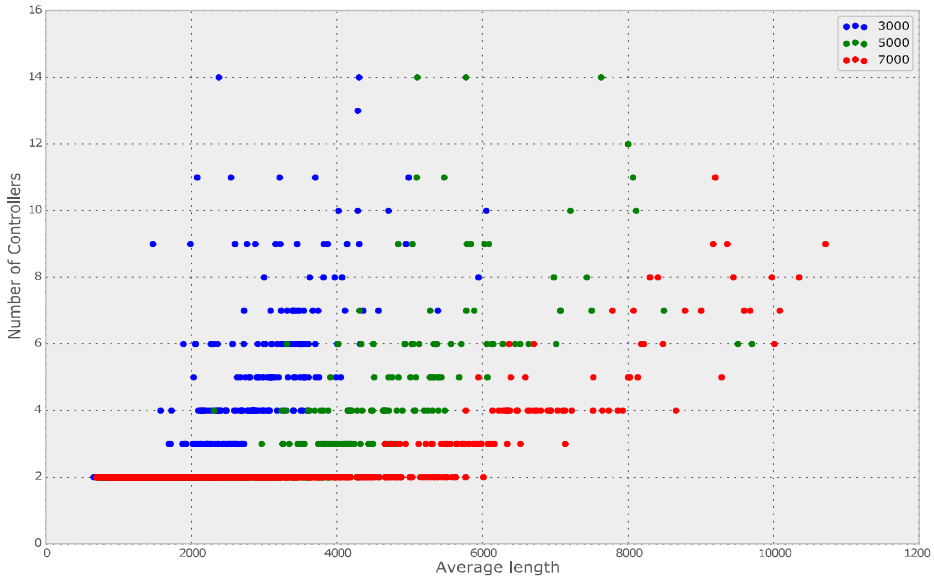


Figure 91: Number of controllers on the average length of the shortest path in the graph and on maximal latency

Figure 92 illustrates the increase of required controllers when the graph density decreases. For low values of maximal latency there is no solution under a threshold (around 0.2) of graph density.

5.2.2.4 Conclusion

In this work, we have proposed new formulations based on Integer Linear Programming for the Controller Placement Problem. Specifically, we have proposed a formulation to minimize the number of controllers in a SDN based network architecture. It takes into account the maximal allowed latency between routers and controllers, the maximal allowed latency between controllers, and the load balancing constraint between the controller's sub-domains. This formulation has been extended to deal with resilient SDN architecture. The solutions of this second formulation provide then the optimal number of controllers, their placement within the network, their assigned routers and finally several levels of back-up controllers in case of failures of the primary controllers.

Both random and realistic network instances with up to 80 nodes have been tested and solved to optimality.
In future work, we intend to derive a heuristic procedure based on IP relaxations to deal with world-wide topology.

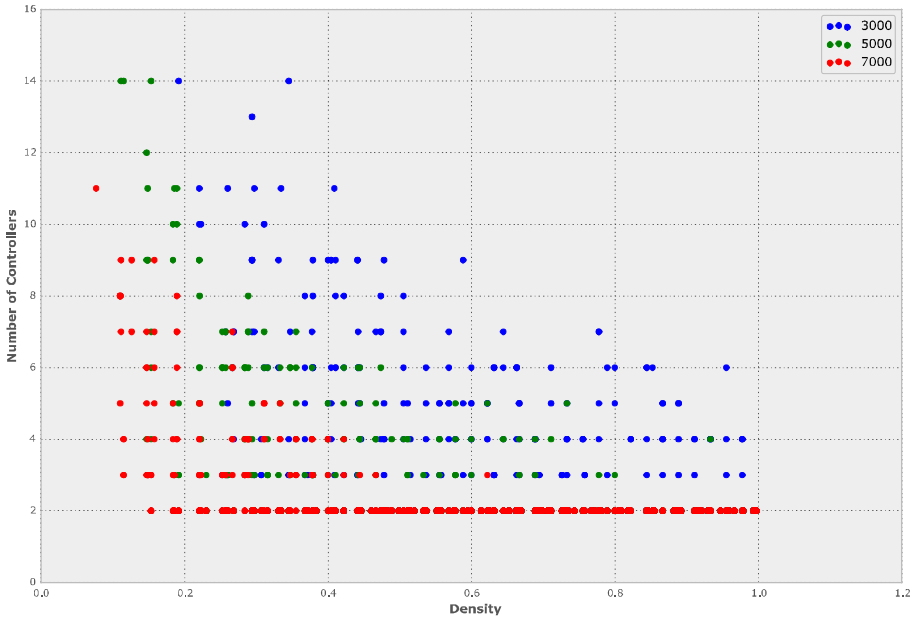


Figure 92: Number of controllers on the graph density and on maximal latency

5.2.3 Geographical load balancing in SDN-based data centres

In this subsection we demonstrate how SDN allows saving energy in a network of Data Centers (DCs). We are looking in particular to answer the question: what are the benefits of a geographical load balancing algorithm between data centers? The SDN controller has a centralized view of the whole system, thus it enables to implement algorithms that could provide an optimized solution over all the data center network.

We consider a network composed of DCs with the following characteristics:

- The DCs are located in different geographical zones, thus in different time zones.
- The DCs generate renewable energy, thanks to photovoltaic panels for instance. The quantity of energy generated depends on the time of the day. In the following, the renewable energy will also be referred as green energy.
- Each DC receives requests for videos, and can either process the request itself, or assign it to another DC. Requests can only be assigned once

Figure 93: Orange Business Together as a Service hosting platform architecture

The figure above shows an example of a DC network, for a connectivity service offered by Orange Business. The DCs are interconnected via high bandwidth VPN.

Energy savings occur thanks to two main leverages: First, the sleep mode of the servers inside the DCs and second the green energy usage, where and when it is available. Our proposed load balancing algorithm makes the most of these two leverages. In order to evaluate the performance of our algorithm, we have developed and solved an explicit formulation of the global optimization problem, so that we know the energy consumption of the best load balancing.

The remainder of this subsection is organized as follows. In Section 5.2.3.1 we give the problem statement and its formulation as a multi-period optimization problem. The on-line algorithms are then described in Section 5.2.3.2. The numerical results are presented and discussed in Section 5.2.3.3, and finally, Section 5.2.3.4 concludes the stufy by presenting some perspectives of this work.

5.2.3.1 The multi period Geographical Load Balancing Problem

In this section, we give a formal description of the Multi-Period Geographical Load Balancing (MPGLB) problem for which we propose a formulation.

Characteristics of the problem and notations

The data-center network is modeled as a digraph G = (V, E), where the set of nodes V represents the DCs and the set of edges E their direct interconnection links. An interconnection link (i, j) ∈ E corresponds to a physical path between two data centers i and j and its capacity, denoted cij, corresponds to the minimum of all residual capacities along the physical path. It is expressed in terms of number of requests. Let Pij be the set of links used along the path to connect DC i to DC j. As we consider only video services the delay is not as critical as for interactive services: as long as the delay is under an acceptable value for the end-user who requested a video, it is not necessary to minimize it. Furthermore, we assume that all generated paths Pij respect the required delays, thus the delay constraints are relaxed from the model.

We assume that the evolution of traffic and of green energy is periodic, typically a daily period. This period is decomposed in homogeneous time slots t ∈ {0, … , T}, and $\boldsymbol{o_i}$ is the time offset of DC i. The video requests that are considered in this study are characterized by the default attached DC i of the request, and by their duration in number of time slots. Several classes c ∈ 1, …, C of video requests are defined, sorted by increasing order of their duration D = {d1, …, dc, …, dmax}, where dmax is the duration of the longest video. Then, the number of requests of class c sent by users attached to DC i during time slot t is denoted $\boldsymbol{S_{ic}^t}$.

Considering hardware equipment in each Data Center, we assume that the physical server characteristics are homogeneous, with a unique capacity M, corresponding to the number of Virtual Machines (VM) that can run simultaneously on a server.

The number of servers in DC i is denoted $\boldsymbol{Mi}$.

To define the energy consumptions, we use the following notations:

- $\boldsymbol{E_i(t)}$ energy consumption of the non-IT infrastructure of DC i during time slot t (in W.h),
- $\boldsymbol{G_i(t)}$ green energy produced by DC i in the time slot t (in W.h),
- $\boldsymbol{c_{VM}}$ energy consumption of the creation of a VM (in W.h),
- $\boldsymbol{e_{VM}}$ energy consumption of the execution of a VM during a time slot (in W.h),
- $\boldsymbol{e_{PR}}$ energy consumption of a physical server running during a time slot (in W.h),
- $\boldsymbol{e_{PS}}$ energy consumption of a physical server in sleep mode during a time slot (in W.h),
- $\boldsymbol{e_P}$ energy consumption for a physical server to go from sleep mode to running mode (in W.h),
- $\boldsymbol{e_{ij}}$ energy consumption of sending a request from DC i to DC j (in W.h).

The energy consumption of the non-IT infrastructure of DCs is characterized by a fixed part (for instance light and fans) and a part that depends linearly on the number of requests processed (for instance the air cooling). As a consequence, we define Ei(t) as:

$$E_i(t) = E_i^{\min} + \left(E_i^{\max} - E_i^{\min}\right)\left(\frac{R_i^t}{M_i M}\right)$$

where $\boldsymbol{R_i^t}$ is the number of requests to process, $\boldsymbol{E_i^{min}}$ the consumption with no request, and $\boldsymbol{E_i^{max}}$ the consumption at full load.

The green energy production is based on a solar energy model and therefore depends on the local time of the day. It is defined as:

$$G_i(t) = G_i G[(t - o_i) \bmod 24]$$

where G_i is the total daily solar energy produced by DC i, and G(t) a normalized function so that $\sum G(t)$ = 1.
The number of requests sent by users is based on a daily traffic profile given by a function L(t). This function is normalized so that $\sum L(t)$ = 1. As a consequence, we can write

$$S_{ic}^t = \alpha_c S_i L[(t - o_i) \bmod 24]$$

with ∝c the proportion of requests of class c, and Si the total daily number of requests sent by end-users to DC i.

Mixed Integer Linear Programming Formulation

We consider the following sets of variables: ∀t ∈ {0, …T}, ∀i ∈ {0, …|V|},

- x_i^t the number of VMs at time t in DC i,
- x'^t_i the number of VMs created at time t in DC i,
- y_i^t the number of running physical servers at time t in DC i,
- y'^t_i the number of physical servers leaving sleep mode at time t in DC i,
- λ_{ij}^t the number of requests of class c sent from DC i to DC j at time t,
- R_{ic}^t the number of requests of class c processed locally at DC i at time t,
- R_i^t the number of requests processed locally at DC i at time t,
- Fi(t) the non-green energy consumed by DC i at time t (these variables are introduced for the linearization of the problem)

Then, a formulation for the Multi-Period Geographical Load Balancing Problem (MP-GLBP) is:

$$min \sum_{i=1}^{|V|} \sum_{t=1}^{T} F_i(t) + \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \sum_{c=1}^{C} \sum_{t=1}^{T} \lambda_{ijc}^t e_{ij} \qquad \text{(OBJ)}$$

such that

$$R_{ic}^t = S_{ic}^t + \sum_j \lambda_{jic}^t - \sum_j \lambda_{ijc}^t \qquad \forall i, \forall c, \forall t \quad (1)$$

$$R_i^t = \sum_{a=0}^{min(d_{max},t)-1} \sum_{d_c > a} R_{ic}^{t-a} \qquad \forall i, \forall t \quad (2)$$

$$\sum_j \lambda_{ijc}^t \leq S_{ic}^t \qquad \forall i, \forall c, \forall t \quad (3)$$

$$R_i^t \leq x_i^t \leq y_i^t M \leq M_i M \qquad \forall i, \forall t \quad (4)$$

$$x_i^t - x_i^{t-1} \leq x_i'^t \qquad \forall i, \forall t > 1 \quad (5)$$

$$y_i^t - y_i^{t-1} \leq y_i'^t \qquad \forall i, \forall t > 1 \quad (6)$$

$$x_i^t e_{\text{VM}} + x_i'^t c_{\text{VM}} + y_i^t e_{\text{PR}} + y_i'^t e_P + \left(M_i - y_i^t\right) e_{PS} + E_i(t) - G_i(t) \leq F_i(t) \qquad \forall i, \forall t \quad (7)$$

$$0 \leq F_i(t) \qquad \forall i, \forall t \quad (8)$$

$$\sum_{a=0}^{min(d_{max},t)-1} \sum_{d_c > a} \sum_{st.(ij) \in P_{IJ}} \lambda_{ijc}^{t-a} \leq c_{ij} \qquad \forall (ij) \in E, \forall t \quad (9)$$

Constraints (1) express the fact that the number of requests processed at each time in a DC must be equal to the number of requests sent by end-users, plus the requests sent from other DCs, minus the requests sent to other DCs, for each class of requests. Constraints (2) express the fact that the number of requests processed must be equal to the number of requests processed at time t, plus the number of requests processed at previous times that are still running. It also means that requests are not moved once they have started to be processed by a VM. Constraints (3) ensure that requests can be migrated only once. (4) ensure that each request is processed by a VM, and that each VM is hosted on a physical server. (5) and (6) ensure that VMs are created and physical servers are turned on when necessary. (7) and (8) come from the fact that green energy can only be used locally where it is produced, and it cannot be stored to be used at a later time. Finally, (9) are capacity constraints to ensure that all the traffic respects the capacity limit of each link. In (9) we have to sum the requests migrated at the current time and also from the previous times for the requests that last more than one time slot (hence the sum over the index a). Moreover, we have to sum on all the (pre-computed) paths that contain the edge (ij).

Geographical Load Balancing Period by Period (GLBPP)

By decomposing the MP-GLB problem for each period, we obtain T sub-problems, the so-called GLBPP problems. Each one gives rise to the optimal mapping of the requests to the data centers when we have no knowledge of the future requests. This more realistic variant is used to prove the efficiency of the results obtained with the on-line algorithms described in the next section.

5.2.3.2 On-Line Algorithms

In this section, we present two on-line algorithms that could be implemented in a real-world setting. An on-line algorithm is implemented in a Load Balancer (LB), which is itself a component of the SDN controller. The LB is therefore located in a logical central point of decision.

We now consider a continuous time and requests arriving according to a Poisson law (i.e. with exponential inter-arrival times). Each request is characterized by an arrival time and a duration (which is the duration of the requested video).

The on-line algorithm, upon the arrival of each request, decides where it is going to be processed. The first algorithm is called LLB (Local Load Balancing) and it performs local assignment, i.e. each request is assigned to the local DC (recall that each request is attached to a default DC). LLB is used for reference in order to estimate the benefits of request migration. The second on-line algorithm is called GGLB (Green Geographical Load Balancing). It is based on the computation of the migration reward for each request.

We introduce the following additional notations

- PVM the power draw of a VM (in Watt (W)),
- PPR the power draw of a running physical server (in W),
- PPS the power draw of a physical server in sleep mode (in W),
- Pi(R) the power draw of all the non-IT infrastructure of DC i, with R requests to process (in W),
- E1i (d; t) energy consumed by DC i for the treatment of 1 request of duration d at time t (in W.h),
- C1i (d; t) non-green energy consumed by DC i for the procesing of 1 request of duration d at time t (in W.h),
- PGi (t) green power available at DC i, at time t,
- MR(d; t; i; j) (for Migration Reward), the non-green energy saved if a request of duration d is migrated at time t from DC i to DC j.

The on-line algorithm GGLB

When a request of duration d arrives at time t at the DC i, the load balancer computes the MR(d; t; i; j) for all the DCsj ϵ {1, …, |V|}. Note that for j = i, MR(d; t; i; i) = 0. MR(d; t; i; j) represents the potential gain of migrating the request from DC i to DC j, therefore it is written as

$$\mathrm{MR}(d,t,i,j) = C_i^1(d,t) - [e_{ij} + C_j^1(d,t)],$$

i.e. the cost of local processing minus the cost of processing at remote DC j augmented by the cost of transit (the cost is expressed in terms of non-green energy consumption). The cost of processing a request can be computed thanks to the algorithm below. In a first step, the algorithm computes the energy consumption E1i (d; t), and then it computes the nongreen energy consumption.

The load balancer then assigns the request to the DC j with the highest MR(d; t; i; j). In order to evaluate the performance of on-line algorithms, we have designed and implemented a simulation tool, that is described below.

Algorithm 1 Computation of $C_i^1(d,t)$, incremental energy cost of processing a request

1: Computation of $E_i^1(d,t)$

1) If at least one physical server is running and not fully loaded:

$$E_i^1(d,t) = P_{\text{VM}}d + C_{\text{VM}} + P_i(1)d$$

2) If all the running servers are fully loaded:

a) if there is a server in sleep mode, then E_i^1 is the same as above, augmented by the following amount: $C_P + (P_{\text{PR}} - P_{\text{PS}})d$

b) If there is no more available server, then we set the cost as infinite, since this DC cannot process the request $E_i^1(d,t) = +\infty$

2: Computation of $C_i^1(d,t)$, knowing $P_i^G(t)$

1) If the green power is sufficient to process the request, then $C_i^1(d,t) = 0$

2) If the green power is not enough, then

$$C_i^1(d,t) = E_i^1(d,t) - P_i^G(t)d$$

5.2.3.3 Results and Discussion

Input values

In order to obtain numerical results for large instances, we have generated random instances. The random values of the parameters indicated in the table below are generated with a Gaussian distribution. Some of the parameters are kept as fixed values. We have used two classes of requests, one class of duration 1 hour (80% of the requests) and one class of duration 2 hours (20%). The parameters oi (timezone offset) are determined randomly with a uniform distribution in {0, …, 23}. Parameters Si have random values according to a Gaussian distribution, and, by varying its average and its variance, we define four types of instance, as shown in the table below. Finally, the parameters Mi are set so that the DC i would reach a load of 80% at the traffic peak. For each type of instance, we have generated random instances by varying numbers of DC, from 10 to 70.

Table 25: Random value inputs (in KW.h)

Input	$E_i^{\min}$	$E_i^{\max}$	e_{ij}	G_i
Average	1.5	3	2.10^{-5}	500
Variance	0.3	0.6	10^{-5}	300

Table 26: Fixed value inputs

Input	$\|V\|$	C_{VM}	e_{VM}	e_{PR}	e_{PS}	c_P	α_1	α_2
Value	100	10	6	300	50	100	80	20
Unit	nb	W.h	W.h	W.h	W.h	W.h	%	%

Table 27: Random inputs

Type of instance	A	B	C	D
Average S_i	15,000	30,000	60,000	150,000
Variance S_i	7,000	15,000	30,000	70,000

Figure 94 shows the traffic load functions L(t) and the solar energy production G(t), as defined in section II-A. The function L(t) was measured on an operational VoD (Video on Demand) service, over a 1-year period. We can observe that the peak of traffic occurs in the early evening, while the peak of solar energy production obviously occurs around noon.

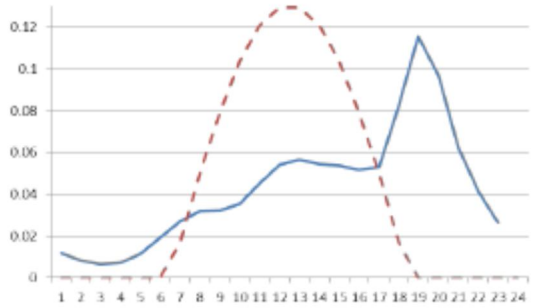


Figure 94: Normalized functions L(t) (plain) and G(t) (dotted)

Implementation details

The MP-GLB and GLBPP problems have been implemented respectively in Java and in Python, thanks to the Pyomo library. Both problems are solved to optimality using the solver CPLEX.

The simulation tool in which algorithms LLB and GGLB run was implemented by us in Python. The simulation generates requests with exponential inter-arrival times, and the rate is determined such that the average daily number of requests is equal to Si. The simulation tool manages and keep track of the state of each DC at all time, with the request assignment determined by the load balancing algorithm considered.

Results

The computational studies are made on a computer with 16 Xeons at 2.13GHz and 32MB of RAM. The largest instance, with 70 data centers, is solved within less than 20 seconds for the MP-GLB problem and less than 6 seconds for the GLBPP problem.

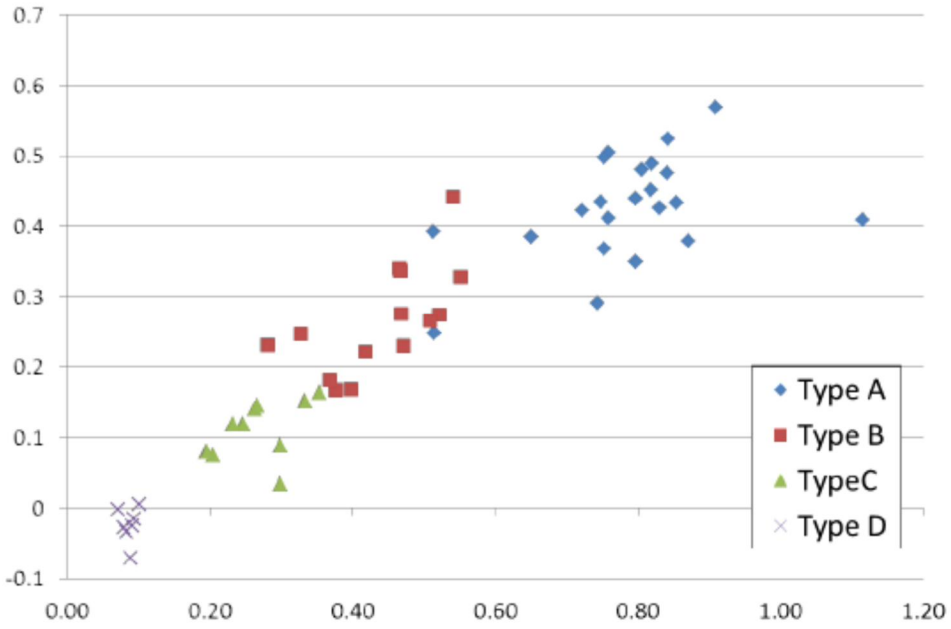


Figure 95: Gain of GGLB (x-axis) vs. γ (rel. green energy production)

Figure 95 above shows the gain of Geographical Load Balancing (in Y-axis) versus γ, the relative green energy total production (in X-axis). The gain of GGLB is computed as the relative difference of GGLB minus LLB, for each instance. The green energy is expressed as the ratio of the total green energy produced by all the DCs, by the total energy consumed by the DCs. Each point on the figure represents a random instance, for which the number of DCs varies on a range from 10 to 70. Please note that when is equal to 1 (or greater), then green energy is necessarily sufficient to process all the requests locally, because green energy is not necessarily available at the time requests arrive, thus resulting in lost green energy. It is interesting to notice the linear trend on the figure. It shows that the more green energy is available, the more beneficial is geographical load balancing. Another striking fact is that for lower values of green energy, the gain becomes negative, which means that it costs more to assign requests to remote DCs than locally.

Table 28: Summary of results

Type of instance	γ	LLB gap to GGLB (rel.)	GLB gap to MP-GLB (rel.)	GLBPP gap to MP-GLB (rel.)
A	0.783	0.426	0.385	0.025
B	0.440	0.265	0.352	0.052
C	0.269	0.112	0.255	0.038
D	0.087	-0.025	0.043	0.017

Table 28 above sums up the results obtained for each instance type. The second column gives the values of the average. The third column gives the average relative gap of LLB to GGLB. It represents the relative gain of assigning requests to any DC, as opposed to only locally. The fourth column is the average relative gap of GGLB to MP-GLB. It represents the gap of our on-line algorithm to the optimal request assignment. Finally, the last column is the average relative gap of GLPPP to MP-GLB. It represents the gain of having a good knowledge of the future requests, as opposed to assigning the requests period by period.

When the green energy production is close to zero, the algorithm becomes counter-productive. This is due to the fact that there is a cost to assigning a request to a distant DC. The GGLB algorithm sometimes chooses a remote DC in order to exploit the green energy produced there, but the

production is so small that it would have been used anyway by the local requests. In that particular case, assignment to a distant DC is not desirable. It must also be noted that, in that case, the gap to the optimal assignment is 4.3%, which means that meaningful gains are not possible anyway. However when is high, the gain of our on-line algorithm is clear, and it goes up to 42%, when the green energy covers 78% of the energy needs. In that case, the gap of our algorithm to optimality is around 38%. This means that, even with the high gains we obtained, the assignment algorithm still can be improved.

Finally, the results show that knowing the future does not bring great benefits, the results are only improved by 2-5%, depending on the instance type. This means that using an on-line with prediction capabilities should not improve significantly its performance.

5.2.3.4 Conclusion and Perspectives

In this work we have provided an efficient on-line algorithm to balance the video requests between data centers, so that the brown energy consumption is minimized. We have also proposed an integer linear programming formulation for the multi period geographical load balancing problem. This formulation, and the formulation obtained by decomposing the problem for each period, have been solved to optimality to prove the good quality solutions of our on-line algorithm. We have made simulations based on real traffic traces and on realistic instance parameters. These numerical results show that we could obtain a gain up to 42% of brown energy reduction thanks to smart load balancing in data center networks. This gain is achieved when the green energy production is high, which may not be the case as of today, but in the future we expect that renewable
energy will be more prevalent.

Since turning on and off servers too frequently can lead to premature wear and tear, we will design an on-line algorithm that requires awaking each physical server a limited number of times per day. We will evaluate how this new constraint reduces the performance of GGLB.

In addition, while the performance of the proposed online algorithm is very good, it is still quite far from the optimal assignment, we will therefore investigate different assignment strategies that could lead to improved performance. Finally, this work could be easily extended to other services by restricting the delay bound while generating the paths related to the service.

6 CONCLUSIONS

Anatomy of end-to-end energy usage for video delivery and its implications for video content generation

We propose an end-to-end energy model and use it in two scenarios, differentiated by the coding format of the video contents. For a sample video encoded with H.264/AVC and H.265/HEVC, we build its energy profile based on measurements conducted separately for encoding and decoding. We find that, neither coding scheme always lead to the lowest energy usage and the best choice is dependant on the popularity of the video. The results also have implications for future research as we showed that the WiFi energy component is negligible compared to the decoding component. Also encoding drains an amount of energy much larger than both the WiFi transfer and the decoding. The concern escalates for battery-operated devices. Therefore, it is important for future research to focus more on the encoding and decoding aspects. Second, the idle energy draw of a device is multiple orders of magnitude larger than the WiFi interface for traffic transmission and reception. This implies that it is important that future research focus on lowering the base power of user device.

Energy-efficient placement of user generated content (UGC)

While UGC is projected to make up a significant share of Internet traffic, we note that very little research attention has been dedicated to this type of content, particularly, when placed in the context of content delivery networks. This study is thus viewed as a first contribution to explore efficient solutions for the placement and delivery of UGC. To this aim, we develop a formulation of the UGC placement problem in a network setting comprising multiple ISPs. Also, as our main contribution in this work, we propose an online algorithm which relies on a valuable attribute of UGC, namely, its strong tie with social networking contexts, to determine the content demand and use it for efficient placement decisions. We advocate our choice of an online technique by recalling that UGC has uncertain persistence, and thus instant decisions must be made upon its exposure to the system. We showed that our proposed online solution performs close to an offline placement solution as a benchmark, while it imposes little or no overhead in terms of inter-ISP coordination and information exchange.

Collaborative content replication and request routing

We have developed an optimization problem for minimum delay content management problem and proposed popularity-based distributed algorithms to approach optimal content replication and request routing policies. Our results, quantified by a delay-relevant metric, reveal that the proposed methods perform close to optimal in the most critical real-world scenarios characterized by a small cache size and long tail popularity distribution attributed to user-generated contents. In non-critical scenarios, the algorithms demonstrate a reasonable accuracy and low delay. It is also highlighted that in scenarios with a long tail popularity distribution, optimizing the software aspect of CDNs may not be sufficient for a quality user experience, and thus infrastructure capacity must be enhanced, too. We also showed, by numerical study, that our proposed cooperative algorithms achieve outstanding power efficiency ratios ranging from 60% to beyond 80%. Possible future work directions include the extension of the model to account for processing delay in the servers and also an extension to proactive replication policies.

Trade-offs in energy usage and server stability in content delivery data centres

We have addressed the challenge of load dispatching in data centres with the requirement of preserving time-varying queue stability and saving of energy-usage, simultaneously. To this end, we have developed a stochastic optimization problem, and derived an algorithm analytically. We show that our proposed algorithm, SEOL, can simultaneously reduce energy usage and queue length of servers. Furthermore, SEOL can offer a desired trade-off between queue length and energy usage by properly tuning a single parameter (V). Specifically, we showed that when the data centre load is high (but below the data centre capacity), setting V to small values leads to shorter queue length compared to the existing load balancing techniques. For high values of the tuning parameter V, a substantially higher gain of energy saving is achieved while the queue stability is also preserved. For future work, we will extend of analysis to obtain an optimality bound for our proposed algorithm. Also, we will address more heterogeneous cases with different server capacities and non-stationary job arrivals.

Fair and efficient resource sharing in interconnected CDNs

Our study of cost (or utility) sharing among multiple ISPs participating in content distribution reveals that efficient mechanisms exist to reach a trade-off between the total cost saved and the share of resources (e.g. bandwidth) allocated to individual ISPs. In particular, we showed that NBS has a desirable potential to achieve such a trade-off. In addition, NBS (or similar bargaining techniques) is simple and parsimonious which makes it easy to implement as a third party component responsible for resource sharing between ISPs. This approach can be employed in the IETF CDN Interworking (CDNI) framework to support efficient interworking among multiple (Telco) CDNs.

CCN and in-network caching

We have proposed a new model for saving energy in content-oriented networks by selectively disabling network elements. We have considered a caching object approach, and jointly optimized the routing and caching object problem. Our model has been validated by solving instances based on real network topologies. Also, using several network performance metrics, we have shown the impacts and the gains of introducing energy-awareness on a real telecommunication network. Furthermore, we have proposed a heuristic which has allowed us to show the benefits of our model compared to the classical routing on shortest path-based caching model.

Resource aware and web-based distributed video delivery

We proposed a web-based distributed video delivery solution based on a hybrid architecture, where in addition to the traditional content server(s) or CDN, video content can also be retrieved from peer mobile devices using WebRTC technology. Video data is divided into smaller segments, which can be downloaded in parallel from both servers and other peers. Resource-awareness is employed for selecting the optimal nodes for downloading the data in order to optimize performance and energy consumption.

Leveraging social information for enhanced content delivery

The proposed idea in this study, namely SocialiVideo, addresses the issue of optimizing content delivery over the Internet by utilizing end-user resources. The main objective of this work is to provide the possibility to users to serve or watch a video from their social connections if both are located in the same geolocation (e.g. Network, City, and Country). This approach is deemed to reduce access delay, perform fast transmission, minimize network load, provide low cost solution for content providers and CDNs, and allow controlling the network with fewer management strategies by the ISPs. In addition, the proposed approach is a promising solution to reduce the power consumption in data centers as well as in service operator side. As recently video dissemination becomes the most popular among social network users, the contribution of this study can be integrated into large-scale social networks such as Facebook, to allow users to gain better interactions and to deliver better services to them. Alternatively, SocialiVideo can be deployed stand-alone especially for some enterprise networks or can be combined with existing CDNs and data centers to enhance their data delivery. As an example, our approach can be integrated with Facebook CDN to deliver most popular content to users. In addition, some locally popular content can be disseminated easily without using any CDN support. Facebook can offer storage devices such as set-top boxes to their customers to keep and serve the content locally.

Energy-efficient cloud design

Our solution based on an orchestrator and a SDN controller reveals that the increase per component is relative low compared to the load of the platform. This indicates that it is more power efficient to utilize as much capacity as possible of a node once it is switched on.

Service function chaining in SDN

We have addressed the problem of optimal deployment of service function chains in terms of maximal energy savings on a network by jointly computing paths and function installation. To this aim, we have developed an optimization framework to solve the service chaining problem with minimal active resources in terms of network edges and nodes. To cope with the time complexity of obtaining the optimal solution, we proposed lightweight heuristic algorithms and showed their performance. We further investigate this problem by exploring the exact procedure(s) to obtain high performance solutions with controlled time complexity.

Controller placement in SDN

We have proposed new formulations based on Integer Linear Programming for the controller placement problem. More specifically, we have proposed a formulation to minimize the number of

controllers in a SDN-based network architecture. It takes into account the maximal allowed latency between routers and controllers, the maximal allowed latency between controllers, and the load balancing constraint between the controller's sub-domains. This formulation has been extended to deal with resilient SDN architecture. The solutions of this second formulation provide the optimal number of controllers, their placement within the network, their assigned routers and finally several levels of back-up controllers in case of failures of the primary controllers.
Both random and realistic network instances with up to 80 nodes have been tested and solved to optimality. In future work, we intend to derive a heuristic procedure based on IP relaxations to deal with world-wide topologies.

Geographical load balancing in SDN

We have provided an efficient on-line algorithm to balance the video requests between data centers, so that the brown energy consumption is minimized. We have also proposed an integer linear programming formulation for the multi period geographical load balancing problem. This formulation, and the formulation obtained by decomposing the problem for each period, have been solved to optimality to prove the good quality solutions of our on-line algorithm. We have made simulations based on real traffic traces and on realistic instance parameters. These numerical results show that we could obtain a gain up to 42% of brown energy reduction thanks to smart load balancing in data center networks. This gain is achieved when the green energy production is high, which may not be the case as of today, but in the future we expect that renewable energy will be more prevalent.
Since turning on and off servers too frequently can lead to premature wear and tear, we will design an on-line algorithm that requires awaking each physical server a limited number of times per day. We will evaluate how this new constraint reduces the performance of GGLB.
In addition, while the performance of the proposed online algorithm is very good, it is still quite far from the optimal assignment, we will therefore investigate different assignment strategies that could lead to improved performance. Finally, this work could be easily extended to other services by restricting the delay bound while generating the paths related to the service.

7 BIBLIOGRAPHY

[1] N. Xu, J. Yang, M. Needham, D. Boscovic and F. Vakil, “Toward the green video CDN.,” in *IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, 2010.

[2] L. Chiaraviglio and I. Matta, “Greencoop: cooperative green routing with energy-efficient servers.,” in *International Conference on Energy-Efficient Computing and Networking*, 2010.

[3] V. Mathew, R. K. Sitaraman and P. Shenoy, “Energy-aware load balancing in content delivery networks.,” in *INFOCOM*, 2012.

[4] C. Ge, N. Wang and Z. Sun, “Optimizing server power consumption in cross-domain content distribution infrastructures,” in *International Conference on Communications (ICC)*, 2012.

[5] V. Mathew, R. Sitaraman and P. Shenoy, “Energy-efficient content delivery networks using cluster shutdown,” in *International Green Computing Conference (IGCC'13)*, 2013.

[6] C. Ge, N. Wang and Z. Sun, “Energy-aware data center management in cross-domain content delivery networks,” in *Conference on Green Communications (GreenCom)*, 2013.

[7] V. Valancius, N. Laoutaris and L. Massoulié, “Greening the internet with nano data centers.,” in *Fifth international conference on Emerging networking experiments and technologies*, 2009.

[8] U. Lee, I. Rimac and V. Hilt, “Greening the internet with content-centric networking,” in *International Conference on Energy-Efficient Computing and Networking*, 2010.

[9] U. Lee, I. Rimac, D. Kilper and V. Hilt, “Toward energy-efficient content dissemination,” *Networks,* vol. 25, no. 2, pp. 14-19, 2011.

[10] K. Guan, G. Atkinson, D. C. Kilper and E. Gulsen, “On the energy efficiency of content delivery architectures.,” in *International Conference on Communications Workshops (ICC)*, 2011.

[11] N. Choi, K. Guan, D. C. Kilper and G. Atkinson, “In-network caching effect on optimal energy consumption in content-centric networking,” in *IEEE International Conference on Communications (ICC)*, 2012.

[12] C. Jayasundar, A. Nirmalathas, E. Wong and C. A. Chan, “Energy efficient content distribution for VoD services,” in *Optical Fiber Communication Conference*, 2011.

[13] Huang, C., Wang, A., Li, J., & Ross, K. W., "Measuring and evaluating large-scale CDNs," *ACM IMC,* 2008.

[14] K. P. Gummadi, S. Saroiu and S. D. Gribble, “King: Estimating Latency Between Arbitrary Internet End Hosts,” in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, New York, NY, USA, 2002.

[15] R. e. a. Krishnan, “Moving beyond end-to-end path information to optimize CDN performance,” in *ACM SIGCOMM*, 2009.

[16] S. P. Narayanan, Y. S. Nam, A. Sivakumar, B. Chandrasekaran, B. M. Maggs and S. G. Rao, “Reducing Latency Through Page-aware Management of Web Objects by Content Delivery Networks,” in *ACM SIGMETRICS*, 2016.

[17] M. Yu, W. Jiang, H. Li and I. Stoica, “Tradeoffs in CDN designs for throughput oriented traffic.,” in *Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM*, 2012.

[18] Z. Akhtar, A. Hussain, E. Katz-Bassett and R. Govindan, “DBit: Assessing Statistically Significant Differences in CDN Performance.,” *Computer Networks,* 2016.

[19] K. Stamos, G. Pallis, A. Vakali and M. D. Dikaiakos, “Evaluating the utility of content delivery networks,” in *UPGRADE-CN workshop on Use of P2P, GRID and agents for the development of content networks. ACM*, 2009.

[20] G. Pallis, “Improving content delivery by exploiting the utility of CDN servers,” *Springer Berlin Heidelberg,* 2012.

[21] M. Pathan, J. Broberg and R. Buyya, “Maximizing utility for content delivery clouds,” *Springer Berlin Heidelberg,* 2009.

[22] J. Broberg, R. Buyya and Z. Tari, “MetaCDN: Harnessing ‘Storage Clouds’ for high performance content delivery.,” *Journal of Network and Computer Applications,* pp. 1012-1022, 2009.

[23] Pathan, M.; Buyya, R., “A utility model for peering of multi-provider content delivery services.,” in *Local Computer Networks, IEEE 34th Conference*, 2009.

[24] M. T. Diallo, F. Fieau, E. Abd-Elrahmane and H. Afifi, “Utility-based Approach for Video Service Delivery Optimization,” in *ICSNC*, 2014.

[25] V. Mathew, R. K. Sitaraman and P. Shenoy, “Energy-aware load balancing in content delivery networks,” in *INFOCOM*, 2012.

[26] J. P. Mulerikkal and I. Khalil, “An architecture for distributed content delivery network.,” in *ICON*, 2007.

[27] S. Triukose, Z. Wen and M. Rabinovich, “Measuring a commercial content delivery network.,” in *Proceedings of the 20th international conference on World wide web*, 2011.

[28] J. Kangasharju, J. Roberts and K. W. Ross, “Object replication strategies in content distribution networks.,” *Computer Communications,* pp. 376-383, 2002.

[29] H. Qian, W. Muqing, W. Dongyang and G. Song, “Lifetime-based greedy caching approach for content-centric networking,” in *ICT*, 2014.

[30] B. Molina, C. E. Palau and M. Esteve, “Modeling content delivery networks and their performance,” *Computer Communications,* pp. 1401-1411, 2004.

[31] A. Bianco, R. Mashayekhi and M. Meo, “Energy consumption for data distribution in content delivery networks,” in *International Conference on Communications (ICC)*, 2016.

[32] K. Guan, G. Atkinson, Kilper, D. C. and E. Gulsen, “On the Energy Efficiency of Content Delivery Architectures,” in *Communications Workshops (ICC)*, 2011.

[33] J. Araujo, F. Giroire, J. Moulierac, Y. Liu and R. Modrzejewski, “Energy efficient content distribution,” *Computer Journal,* 2015.

[34] S. ul Islam and J. M. Pierson, “Evaluating energy consumption in CDN servers.,” in *International Conference on Information and Communication on Technology.*, Berlin Heidelberg, 2012.

[35] K. L. Johnson, J. F. Carr, M. S. Day and M. F. Kaashoek, “The measured performance of content distribution networks,” *Computer Communications,* pp. 202-206, 2001.

[36] B. Krishnamurthy, C. Wills and Y. Zhang, “On the use and performance of content distribution networks.,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001.

[37] C. Canali, V. Cardellini, M. Colajanni and R. Lancellotti, “Evaluating user-perceived benefits of content distribution networks,” in *Int’l Symp. on Perf. Eval. of Comp. and Telecomm. Sys.(SPECTS)*, 2004.

[38] A. J. Su and A. Kuzmanovic, “Thinking Akamai,” in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, 2008.

[39] I. Poese, B. Frank, B. Ager, G. Smaragdakis and A. Feldmann, “Improving content delivery using provider-aided distance information.,” in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010.

[40] A. Ruhela, R. M. Tripathy, S. Triukose, S. Ardon, A. Bagchi and A. Seth, “Towards the use of online social networks for efficient internet content distribution,” in *Advanced Networks and Telecommunication Systems (ANTS)*, 2011.

[41] S. Scellato, C. Mascolo, M. Musolesi and J. Crowcroft, “Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades,” in *Proceedings of the 20th international conference on World wide web*, 2011.

[42] L. Cui and N. Lu, “SocialStreaming: P2P-assisted streaming in social networks.,” in *Computers and Communications (ISCC)*, 2013.

[43] A. Ruhela, S. Triukose, S. Ardon, A. Bagchi, A. Mahanti and A. Seth, “The scope for Online Social Network aided caching in web CDNs,” in *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*, 2013.

[44] J. Baliga, R. Ayre, K. Hinton and R. S. Tucker, “Architectures for energy-efficient IPTV networks,” in *IEEE Conference on Optical Fiber Communication (OFC 2009)*, 2009.

[45] F. Bossen, B. Bross, K. Suhring and D. Flynn, “HEVC complexity and implementation analysis,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, no. 12, pp. 1685-1696, 2012.

[46] Y. Benmoussa, Boukhobza, S. E. J., Y. Hadjadj-Aoul and D. Benazzouz, “A methodology for performance/energy consumption characterization and modeling of video decoding on heterogeneous soc and its applications,” *Journal of Systems Architecture,* vol. 61, no. 1, pp. 49-

70, 2015.

[47] C. M. Kyung and S. (. Yoo, Energy-aware system design: algorithms and architectures, Springer Science & Business Media, 2011.

[48] E. Nogues, R. Berrada, M. Pelcat, D. Menard and E. Raffin, “A DVFS based HEVC decoder for energy-efficient software implementation on embedded processors,” in *In IEEE International Conference on Multimedia and Expo (ICME)*, 2015.

[49] T. A. da Fonseca and R. L. de Queiroz, “Energy-constrained real-time H. 264/AVC video coding,” in *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[50] Z. Ma, H. Hu and Y. Wang, “On complexity modeling of H. 264/AVC video decoding and its application for energy efficient decoding,” *IEEE Transactions on Multimedia,* vol. 13, no. 6, pp. 1240-1255, 2011.

[51] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs and A. Azcorra, “Per-frame energy consumption in 802.11 devices and its implication on modeling and design,” *IEEE/ACM Transactions on Networking (ToN),* vol. 23, no. 4, pp. 1243-1256, 2015.

[52] M. Ö. Demir, G. K. Kurt and M. Karaca, “An energy consumption model for 802.11 ac access points,” in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014.

[53] “Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020,” 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf.

[54] Z. Wang, L. Sun, C. Wu, W. Zhu, Q. Zhuang and S. Yang, “A joint online transcoding and delivery approach for dynamic adaptive streaming,” *IEEE Transactions on Multimedia,* vol. 17, no. 6, pp. 867-879, 2015.

[55] A. Passarella, “A survey on content-centric technologies for the current Internet: CDN and P2P solutions,” *Computer Communications,* vol. 35, no. 1, pp. 1-32, 2012.

[56] A.-M. K. Pathan and R. Buyya, “A taxonomy and survey of content delivery networks,” Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report, 2007.

[57] S. Borst, V. Gupt and A. Walid, “Distributed caching algorithms for content distribution networks,” in *IEEE INFOCOM*, 2010.

[58] M. Feldman and J. Chuang, “Service differentiation in Web caching and content distribution,” in *IASTED International Conference on Communications and Computer Networks*, 2002.

[59] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn and S. Moon, “Analyzing the video popularity characteristics of large-scale user generated content systems,” *IEEE/ACM Transactions on Networking (TON),* vol. 17, no. 5, pp. 1357--1370, 2009.

[60] “Global Internet Phenomena Report: Africa, Middle East and North America,” [Online]. Available: https://www.sandvine.com/trends/global-internet-phenomena/.

[61] A. Antonopoulos, C. Perillo and C. Verikoukis, “Internet Service Providers vs. Over-the-Top Companies: Friends or Foes?-Short talk,” *ACM SIGMETRICS Performance Evaluation Review,* vol. 44, no. 3, pp. 37-37, 2017.

[62] “AT&T CDN Service.,” [Online]. Available: http://www.business.att.com/enterprise/Service/hosting-services/content-delivery/distribution/.

[63] Level3, “Level3 Content Delivery Network,” [Online]. Available: http://www.level3.com/en/products/content-delivery-network/.

[64] J. Ugander, B. Karrer, L. Backstrom and C. Marlow, “The anatomy of the facebook social graph,” arXiv preprint arXiv:1111.4503, 2011.

[65] R. M. Karp, “On-line algorithms versus off-line algorithms: How much is it worth to know the future?,” in *IFIP Congress (1)*, 1992.

[66] H. J. Böckenhauer, D. Komm, R. Královič and P. Rossmanith, “The online knapsack problem: Advice and randomization,” *heoretical Computer Science,* vol. 527, pp. 61-72, 2014.

[67] M. Cygan, Ł. Jeż and J. Sgall, “Online knapsack revisited,” *Theory of Computing Systems,* vol. 58, no. 1, pp. 153-190, 2016.

[68] Y. Zhou, D. Chakrabarty and R. Lukose, “Budget constrained bidding in keyword auctions and

online knapsack problems," in *17th international conference on World Wide Web*, 2008.

[69] "Neos server: State-of-the-art solvers for numerical optimization," [Online]. Available: https://neos-server.org/neos/.

[70] Masanet, E., "Report to the U.S. Congress on Server and Data Center Energy Efficiency: Public Law," Environmental Protection Agency, 20017.

[71] L. Georgiadis, M. J. Neely and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking,* vol. 1, no. 1, pp. 1-144, 2006.

[72] D. P. Palomar and J. R. Fonollosa, "Practical algorithms for a family of waterfilling solutions," *IEEE transactions on Signal Processing,* vol. 53, no. 2, pp. 686-695, 2005.

[73] S. Manfredi, F. Oliviero and S. P. Romano, "A distributed control law for load balancing in content delivery networks," *IEEE/ACM Transactions on Networking (TON),* vol. 21, no. 1, pp. 55-68, 2013.

[74] AT&T, "AT&T CDN Service," [Online]. Available: http://www.business.att.com/enterprise/Service/hosting-services/content-delivery/distribution/.

[75] Alcatel-Lucent, "Velocix Content Delivery Network," [Online]. Available: http://resources.alcatel-lucent.com/?cid=165199.

[76] W. Jiang, R. Zhang-Shen, J. Rexford and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," *ACM SIGMETRICS Performance Evaluation Review,* vol. 37, no. 1, pp. 239-250, 2009.

[77] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann and B. Maggs, "Enabling content-aware traffic engineering," *ACM SIGCOMM Computer Communication Review,* vol. 42, no. 5, pp. 21-28, 2012.

[78] H. Lee, D. Lee and Y. Yi, "On the economic impact of Telco CDNs and their alliance on the CDN market," in *IEEE International Conference on Communications (ICC)*, 2014.

[79] "Content Delivery Networks Interconnection (CDNI)," Internet Engineering Task Force (IETF), [Online]. Available: https://datatracker.ietf.org/wg/cdni/documents/.

[80] M. Pathan and R. Buyya, "A taxonomy of CDNs," in *Content Delivery Networks*, Springer Berlin Heidelberg, 2008, pp. 33-77.

[81] M. A. Hoque, M. Siekkinen and J. K. Nurminen, "Energy efficient multimedia streaming to mobile devices - a survey," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 1, pp. 579-597, 2014.

[82] M. A. Hoque and e. al., "Dissecting mobile video services: An energy consumption perspective," in *IEEE 14th International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2013.

[83] Y. Liu and W. Mea, "An Energy Perspective of Multimedia Streaming Systems," in *Third International Conference on Cloud and Green Computing (CGC)*, 2013.

[84] C. A. Chan et. al., "Energy efficient delivery methods for video-rich services over next generation broadband access networks," in *IEEE International Converence on Communications (ICC)*, 2011.

[85] N. Thiagarajan et. al., "Who killed my battery?: analyzing mobile browser energy consumption," in *International conference on World Wide Web*, 2012.

[86] K. J. Ma et. al., "Mobile video delivery with HTTP," *IEEE Communications Magazine,* vol. 49, no. 4, pp. 166-175, 2011.

[87] J. K. Nurminen et. al., "P2P media streaming with HTML5 and WebRTC," in *IEEE Conference on Computer Communications Workshops (INFOCOM KSHPS)*, 2013.

[88] P. A. M. Zhao, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon and M. Ponec, "Peer-assisted content distribution in akamai netsession," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013.

[89] M. B. a. Y.Ofek, "End-to-end delay analysis of videoconferencing over packet-switched networks," *IEEE ACM Transactions On Network,* 2000.

[90] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM,* vol. 49, no. 1, p. 101–106, 2006.

[91] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging*

networking experiments and technologies, 2009.

[92] J. Araujo, F. Giroire, Y. Liu, R. Modrzejewski and a. J. Moulierac, “Energy efficient content distribution,” in *IEEE International Conference on Communications (ICC)*, 2013.

[93] M. Mangili, F. Martignon and A. Capone, “A comparative study of content-centric and content-distribution networks: Performance and bounds,” in *Proceedings of IEEE Global Communications Conference (Globecom 2013)*, 2013.

[94] L. Chiaraviglio, M. Mellia and a. F. Neri, “Minimizing isp network energy cost: formulation and solutions,” *IEEE/ACM Transactions on Networking (TON),* vol. 20, no. 2, p. 463–476, 2012.

[95] Y. Carlinet, E. Gourdin, B. Kauffmann and Y. Levene, “Some keys for designing an efficient caching architecture,” Orange Labs, Networks and Carriers, 2011.

[96] “CPLEX,” IBM ILOG, [Online]. Available: http://www.ilog.com/products/cplex/.

[97] E. Santos, C. McLean, C. Solinas and A. Hindle, “How does Docker affect energy consumption? Evaluating workloads in and out of Docker containers,” *CoRR,* vol. abs/1705.01176, 2017.

[98] R. Shea, H. Wang and J. Liu, “Power consumption of virtual machines with network transactions: Measurement and improvements,” in *IEEE INFOCOM*, 2014.

[99] R. Morabito, “Power consumption of virtualization technologies: an empirical investigation,” in *IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, 2015.

[100] R. Morabito, R. Petrolo, V. Loscri, N. Mitton, G. Ruggeri and A. Molinaro, “Lightweight Virtualization as Enabling Technology for Future Smart Cars,” in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2017.

[101] R. Morabito and N. Beijar, “A Framework based on SDN and Containers for Dynamic Service Chains on IoT Gateways,” in *ACM SIGCOMM 2017 1st International Workshop on Hot Topics in Container Networking and Networked Systems*, Los Angeles, 2017.

[102] H. Huang, S. Guo, P. Li, B. Ye and I. Stojmenovic, “Joint Optimization of Rule Placement and Traffic Engineering for QoS Provisioning in Software Defined Network,” *IEEE Transactions on Computers,* vol. 64, no. 12, pp. 3488-3499, 2015.

[103] N. Kang, Z. Liu, J. Rexford and D. Walker, “Optimizing the "one big switch" abstraction in software-defined networks,” in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies (CoNEXT '13)*, 2013.

[104] M. Bouet, J. Leguay and V. Conan, “Cost-based placement of vDPI functions in NFV infrastructures,” in *IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[105] K. Suksomboon, M. Fukushima, M. Hayashi, R. Chawuthai and H. Takeda, “LawNFO: A decision framework for optimal location-aware network function outsourcing,” in *IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[106] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan and T. Wood, “Virtual function placement and traffic steering in flexible and dynamic software defined networks,” in *LANMAN*, 2015.

[107] B. Heller, R. Sherwood and N. McKeown, “The controller placement problem,” in *Proceedings of the First Workshop on HotTopics in Software Defined Networks, ser. HotSDN ’12*, 2012.

[108] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner and P. Tran-Gia, “Pareto-optimal resilient controller placement in sdn-based core networks,” in *International Teletraffic Congress (ITC)*, 2013.

[109] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel and M. Hoffmann, “Heuristic approaches to the controller placement problem in large scale sdn networks,” *IEEE Transactions on Network and Service Management,* vol. 12, no. 1, pp. 4-17, 2015.

[110] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev and P. Tran-Gia, “Specialized heuristics for the controller placement problem in large scale SDN networks,” in *International Teletraffic Congress*, 2015.

[111] G. Yao, J. Bi, Y. Li and L. Guo, “On the capacitated controller placement problem in software defined networks,” *IEEE Communications Letters,* vol. 18, no. 8, p. 1339–1342, 2014.

[112] Y. Jimenez, C. Cervello-Pastor and A. Garcia, “On the controller placement for designing a distributed SDN control layer,” in *IFIP Networking Conference*, 2014.

[113] J. Ros and P. M. Ruiz, “Five nines of southbound reliability in software-defined networks,” in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014.

[114] H. Fallah, A. Sadigh and M. Aslanzadeh, “Covering problem,” *Facility Location, ser. Contributions to Management Science,* p. 145–176, 2009.

[115] C. Swamy and D. B. Shmoys, “Fault-tolerant facility location,” *ACM Transactions on Algorithms (TALG),* vol. 4, no. 4, p. 51, 2008.

[116] S. Guha, A. Meyerson and K. Munagala, “Improved algorithms for fault tolerant facility location,” in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 2001.

[117] Z.-J. M. Shen, R. L. Zhan and J. Zhang, “The reliable facility location problem: Formulations, heuristics, and approximation algorithms,” *INFORMS Journal on Computing,* vol. 23, no. 3, p. 470–482, 2011.

[118] E. W. Dijkstra, “A short introduction to the art of programming,” circulated privately, 1971.

[119] R. W. Floyd, “Algorithm 97: shortest path,” *Communications of the ACM,* vol. 5, no. 6, p. 345, 1962.

[120] S. Orlowski, M. Pioro, A. Tomaszewski and R. Wessaly, “SNDlib 1.0–Survivable Network Design Library,” in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007)*, 2007.