Performance Analysis of Content-Centric and Content-Delivery Networks with Evolving Object Popularity

Michele Mangili[a,b], Fabio Martignon[a,c,*], Antonio Capone[b]

[a] *LRI, Université Paris-Sud, Bat. 650, rue Noetzlin, 91405 Orsay, France.*
[b] *DEIB, Politecnico di Milano, via Ponzio 34/5, 20133 Milano, Italy.*
[c] *IUF, Institut Universitaire de France*

Abstract

The Internet is currently mostly exploited as a means to perform massive digital content distribution. Such a usage profile was not specifically taken into account while initially designing the architecture of the network: as a matter of fact, the Internet was instead conceived around the concept of host-to-host communications between two remote machines.

To solve this problem, Content-Delivery Networks (CDNs) are currently used as a well-established technology to serve content-driven demands through an infrastructure that is not tailored for that purpose. On the other hand, the novel paradigm of Content-Centric Networking (CCN) aims at filling the gap of this misalignment by changing the network-layer protocols, solving the content-distribution problem at its root.

In this paper, we formulate novel optimization models to analyze the performance gains that CDN and CCN can achieve, by reducing the total amount of traffic exchanged through the network. We tackle this problem by adopting a time-varying content popularity evolution model that accurately represents the dynamic behavior of users.

We discover that, in most of the cases, CDN reduces the overall traffic ex-

*Corresponding author, Tel: (+33) 01.69.15.68.16, Fax: (+33) 01.69.15.65.86
Email addresses: michele.mangili@lri.fr (Michele Mangili), fabio.martignon@lri.fr (Fabio Martignon), capone@elet.polimi.it (Antonio Capone)

changed between network nodes, leading to better performance, whereas CCN should instead be preferred in those scenarios where CDN cannot quickly react to popularity evolution. On top of that, we show that very limited benefits can be obtained by changing the cache replacement algorithms. Finally, all our key findings are confirmed by simulation campaigns that further complement this work.

Keywords: Content-Centric Networks, Content-Delivery Networks, Performance Analysis, Optimization.

1. Introduction

Nowadays, the way people exploit the services provided by the Internet is radically changed with respect to the years when the Network was initially designed. As a matter of fact, the evolution of online services, as well as the success of digital multimedia diffusion, both demand for new technologies to turn the Internet into an efficient *content distribution infrastructure* [1, 2, 3].

However, this fundamental change clashes with the original design principles that guided the engineering process of the worldwide network. In particular, it is in conflict with the well known end-to-end principle, according to which intermediate network nodes should be specialized to accommodate basic functionalities such as packet forwarding, whereas application-specific needs should instead be implemented only in the end hosts [4]. Among the direct consequences of this choice, IP routers can nowadays reach the impressive (and still improving) capacity of 921 Tbps [5]; however, this same choice has limited the content-distribution capabilities of Internet, since implementing content-caching functionalities at the network layer is quite demanding [6].

To overcome this limitation, by moving content replicas nearer to the actual consumers location, Content-Delivery Networks (CDNs), such as Akamai [7], are currently used to efficiently serve the content requests of worldwide Internet users, in today's TCP/IP Internet. CDNs are also used to effectively support sudden popularity changes, known as *flash crowds*, which can mine the reliability

of the system by overwhelming the servers with a huge number of requests.

Rather than working at the application layer, keeping IP as network protocol, a different approach is instead supported by innovative designs known under the name of *"Content-Centric Networks"* (CCNs), which are recently gaining momentum in the research community [8, 3]. These designs propose to unleash the content-distribution potentials of the Internet by using novel network protocols. In particular, one of the advantages obtained switching to these designs is that they can be used to easily provide distributed *in-network caching* at the network level: any router can store (and serve) local copies of given data, thus making the content be replicated closer to the locations where most of the users are actually requesting it, without requiring application-layer solutions.

Moved by the desire (and necessity) to understand whether the migration towards CCN can provide significant benefits to network providers, in this paper we present both theoretical and simulated results that analyze and compare the performance of the CCN and CDN architectures. In particular, we focus on their *performance bounds*, in order to find out whether CCN or CDN benefit of a "natural" performance gain accountable to the intrinsic characteristics of the specific distribution architecture. We consider a scenario where *time-varying* content popularity demands are generated by the consumers, in such a way that we can assess the network capability to react to the dynamic content popularity evolution. In order to do so, we formulate novel optimization models to characterize the performance bounds of the CCN and CDN architectures.

Our key findings suggest that in most of the topologies we considered, when the content popularity evolves very quickly, CCN minimizes the overall network traffic, while CDN should instead be preferred whenever the content popularity dynamics evolves at a slower pace. Another take-home message of our work, in line with the results presented in other papers (i.e., [9]), is that it is better to deploy caching storage on a limited number of nodes rather than distributing it uniformly throughout the network. All the results that we obtained are confirmed by a simulation campaign that we performed on different network topologies.

Our main contributions can be summarized as follows:

1. We formulate a novel optimization model to study the performance bounds of a Content-Centric Network, solving the joint object placement and routing problem, under a realistic, *time-varying* object popularity evolution scheme and with the most notable cache replacement policies [10].

2. We formulate an optimization model to represent a similar scenario in a Content-Delivery Network (CDN), where replica servers are distributed according to the *k-median* model [11]. We build the CDN model in a way such that we can control the speed of reaction of the network to content popularity changes.

3. We extend the ndnSIM simulator [12] to represent the time-varying object popularity. We then compare the simulated results and those obtained with the optimization models under the same parameterization, and for different cache replacement policies.

4. We discuss the obtained results, and show that in most of the topologies considered, CCN should be preferred when the content popularity evolves more quickly; in the Netrail topology, for instance, CCN reduces the traffic more consistently than CDN only if this latter is at least 15 times slower to react to popularity changes. We also find out, in some topologies like Géant, that CDN is always to be preferred since it reaches up to 14% better performance than CCN.

Our model formulations capture the dynamics of the parameters that we deemed most important for the overall system performance of both a CCN and CDN system, in order to make a head-to-head comparison between the two architectures.

This paper is structured as follows: Sec. 2 describes the CCN and CDN paradigms and motivates the choices we made to evaluate the content distribution performance of the network. Sec. 3 describes the content popularity

evolution model. Sec. 4 illustrates the proposed optimization models used to study the performance bounds. Numerical results obtained solving these models are presented and discussed in Sec. 5. In Sec. 6 we discuss related works. Finally, concluding remarks are presented in Sec. 7.

2. Evaluating Content Distribution Performance

In this section, we describe relevant characteristics of Content-Centric (Sec. 2.1) and Content-Delivery Networks (Sec. 2.2) to evaluate and compare their content distribution performance. Finally, in Sec. 2.3 we illustrate and motivate the methodology we used to perform such comparison.

2.1. Content-Centric Networks

We first introduce the features provided by Content-Centric Networks (CCN) that are relevant for our study. A comprehensive description of CCN can be found in [8].

In the literature, several network designs such as [13, 14, 15] are presented as "Information-Centric Networks" (ICNs); however, in this paper we focus our attention on the proposal known under the name of "Content-Centric Networking" (CCN) [8], since it is, to the best of our knowledge, the architecture that has so far received most of the attention from the community.

The communication model proposed by CCN replaces IP host addresses with content names: rather than stating the location *where* the data can be found, in CCN nodes declare *what* information they would like to retrieve. CCN has two distinguished packet types: (1) *Interest* and (2) *Data* packets. The former does not contain the actual data, but it only declares that a node is willing to access a given object whose name is known.

The structure of a CCN router is characterized by three tables: (1) the *Pending Interests Table* (PIT); (2) the *Content Store* (CS) and (3) the *Forwarding Information Base* (FIB).

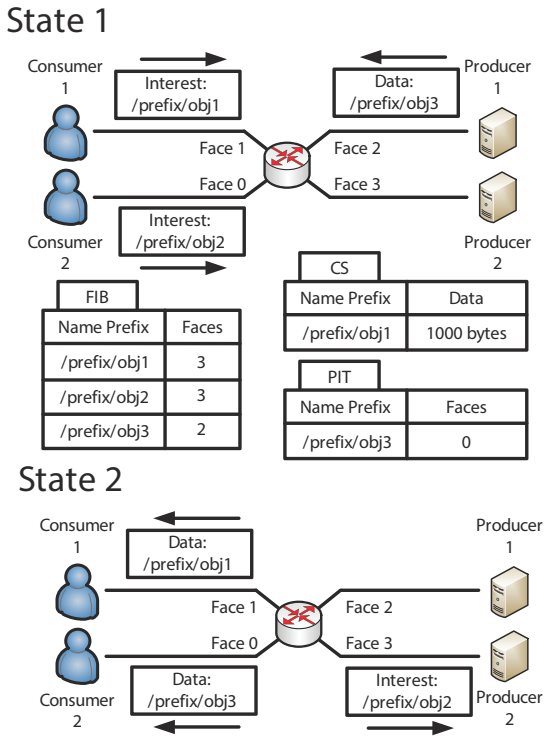


Figure 1: Example illustrating the behavior of a CCN router. Two Interests and one Data packet are received by the router in *State 1*. Given the information contained in the Pending Interests Table (PIT), the Content Store (CS) and the Forwarding Information Base (FIB), in *State 2* the router forwards two Data packets and one Interest.

The PIT is responsible for memorizing the list of Interests previously forwarded, but not yet answered. Interests might arrive from physical hardware interfaces as well as logical applications running on the node itself and called "*faces*". The PIT stores the faces from which Interests were originally received, in order to implement *reverse path forwarding*: as soon as a router receives a Data packet, it checks the PIT and forwards the packet on the same faces from which Interests for that object arrived. The CS is the data structure used to implement *universal in-network caching*. When an Interest arrives, the router initially queries the CS and, in case of a cache hit, it can directly serve the data. The FIB comes into play when a cache miss happens: it contains the

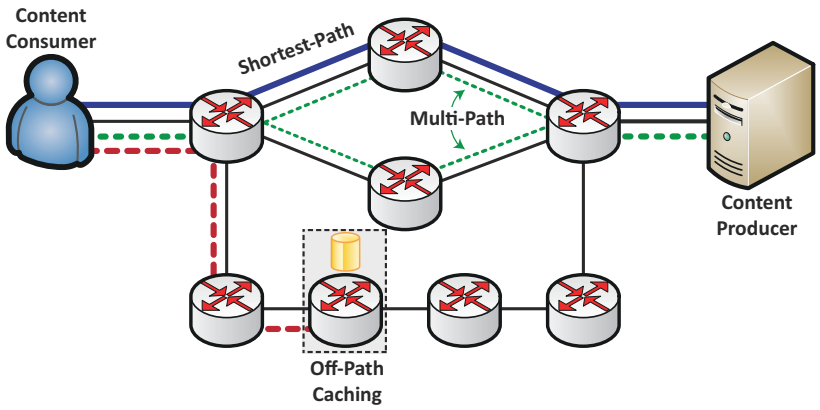


Figure 2: *Multipath Forwarding and Off-Path Caching.* The Figure shows a network where the content consumer retrieves the object directly from the content producer, or from a cache. When retrieving the object from the original producer, the flow is routed on the shortest path. Multipath forwarding functionalities may be used to reduce the link congestion. On the other hand, off-path caching involves the possibility to forward a packet on a path that is not the shortest towards the content producer, but is such that an intermediate cache is storing a copy of the requested content.

next-hop information for prefix names.

An example showing the behavior of a CCN router is depicted in Fig. 1. In State 1, the router receives two Interests and one Data packet. As shown in State 2, the Interest for object `/prefix/obj1` is served by the router since it is available in its CS. The Interest for `/prefix/obj2` will be forwarded to Face 3 since it is the destination available in the FIB. Lastly, the Data packet for `/prefix/obj3` will be forwarded to Face 0, as written in the PIT.

A CCN node may cache only the subset of contents that traversed the node at some point in time, moreover it is reasonable to support the idea that CCN nodes belonging to different autonomous systems will be run by different owners, a feature that makes in-network caching be fully decentralized.

Native support for multipath packet routing is provided by CCN, according to which flows may be split over multiple paths, as shown in the example of Fig. 2. This feature helps mitigating the negative effects of network congestion. On top of that, we also take into account *off-path* caching, as illustrated in Fig. 2:

on-path caching forces flows to be always forwarded on the shortest path to the closest producer publishing the requested content; whereas, *off-path* caching lets network nodes divert traffic requests on a path where a copy of the content can be retrieved from a cache, thus saving the cost to contact the original producer.

2.2. Content-Delivery Networks

A Content-Delivery Network (CDN) is a communication infrastructure composed by a set of machines, known as *surrogate* (or *replica*) servers, geographically distributed in many *Points-of-Presence* (PoPs), to efficiently serve copies of given contents to nearby users [7, 16, 17].

Despite the fact that the CDN infrastructure is massively distributed on a global scale, all the machines belong to a single owner that runs the network with the precise aim to sell the distribution services it offers. Therefore, since the CDN owner controls the whole infrastructure, he is capable to push any content he wants to any node, as well as choose which replica server should satisfy a given content request.

However, at the same time, pushing a content to a given location in the CDN infrastructure involves a computational and transmission overhead that instead has no counterpart in the CCN network. Therefore, while it is theoretically possible to frequently update the content cached on a CCN node, to perform a *fair* comparison, we must take into accurate account the fact that a surrogate server in a CDN will refresh its content catalog at a *much slower rate*, as in [18]. We further assume that the CDN owner has already deployed replica servers by centrally optimizing their placement with standard optimization techniques.

2.3. Methodology to Evaluate the Content Distribution Capabilities of CCN and CDN Architectures

In order to compare the performance of these two types of networks, we have chosen to leverage *offline optimization* techniques and to further confirm the results by performing *simulation campaigns*.

The rationale for using optimization techniques is that we want to study the performance bounds that these architectures can achieve given their intrinsic architectural features, abstracting out all the implementation-specific details. This is done in order to find out whether CCN or CDN can have a "natural" performance gain over the other, accountable to the architectural characteristic of the specific content distribution infrastructure. As extensively discussed in the numerical results section, one of the key findings of our work is that each of these parameters has indeed an impact on the observed system performance.

The main focus of our study is to evaluate the performance of CCN and CDN. We assume that the most realistic case is the one where the object popularity evolves in time, making the traffic demand profile be expressed as an input parameter that is *time-variant*. In our models, time is a discrete quantity expressed as a finite set of time-slots, denoted with $\mathcal{T}$ and such that each $t \in \mathcal{T}$ has a fixed duration. Moreover, we performed such evaluation under different conditions of content popularity evolution (slow/medium/fast evolution speed) to represent the burstiness of object traffic demands.

The performance metric that we take into account is the *total traffic* exchanged in the network, which should be minimized. The models then perform optimal *object placement* and *routing* choices for both the CCN and CDN architectures.

We do not take into account cooperative caching [19, 20] for both our models of a CCN and CDN. As a matter of fact, cache cooperation often introduces significant communication overhead to let the nodes exchange data regarding the set of objects they cache. In other words, in our model for the CCN we assume that each cache behaves autonomously, meaning that they will cache only objects that they were caching, or forwarded in the immediate past. On the other hand, in the CDN model we assume that the network is operated by a single owner capable to optimally place a content at a given node, without the need to make caches explicitly cooperate with each other.

We differentiate our model for a CCN with the one for a CDN by making three assumptions discussed hereafter and summarized in Table 1, where we

Table 1: Modeling Assumptions for the CCN and the CDN.

Assumption	**CCN**	**CDN**	**Ref.**
A1: Storage Size and Placement	Small storage on each router	Large storage on few caches placed with k-median model	[8, 9] [21, 20]
A2: Cache Update Speed	Fast updates are possible	Objects can be updated only after a delay $\lvert\mathcal{T}_r\rvert \in \mathbb{Z}^+$	[8, 18] [20, 22]
A3: Object Pushing	Pull model: a node can cache objects it forwarded or stored in the immediate past	Pull/Push model: any object can be cached at a node	[8, 20] [23]

also provide references to other research works where similar assumptions have been made.

A1: Storage Size and Placement. A given amount of caching storage, denoted with S, is uniformly distributed on all the caching routers $r \in \mathcal{R}$ in the CCN model. On the other hand, in the CDN model, only CDN nodes have caching capabilities. We denote with $\mathcal{D}$ the set of CDN nodes, whose cardinality is restricted to be $N = |\mathcal{D}|$, where $N \leq |\mathcal{R}|$. The same total caching storage is distributed in the two networks, but, due to their lower cardinality, each CDN node will usually store more objects than those persisted in a CCN node. CDN nodes are pre-allocated optimally using a well-known *replica server placement* model. This assumption realistically models a CDN, since its owner will choose to deploy the machines only in the locations where they are mostly useful.

A2: Cache Update Speed. Another clear design requirement that our models meet is that there must be a tunable parameter that lets us change the speed at which the CDN can adapt to sudden popularity changes. We call this parameter "*Relocation Time*" ($\mathcal{T}_r$). In our CDN model, the relocation time is a subset of consecutive time-slots $\mathcal{T}_r \subseteq \mathcal{T}$, and it is used to represent a time window under which CDN nodes cannot change the objects they persist in their storage. By setting $|\mathcal{T}_r| = 1$, we are forcing object relocation in CDN to have the same dynamics of the popularity evolution model. Since we can say that CCN can always promptly react to popularity changes, with $|\mathcal{T}_r| = 1$ CDN is "*as reactive as*" CCN; instead, when we set $|\mathcal{T}_r| = 10$, for instance, it means

that CDN is 10 times slower than CCN.

A3: Object Pushing. A relevant feature that our models take into account is the fact that a CDN is run by a single owner in a centralized manner: at a given point in time the owner can choose to send a replica of a given object to any network node, by pushing it towards that destination, for instance because it performed popularity forecasting and chose to pre-fetch an object on a given server.[1] The same condition does not apply to the CCN model, where instead we restrict the objects that a given router can cache to the subset of those it forwarded in the past. In the numerical results we explicitly study the cost to move the objects to the CDN surrogates, and we show that it has negligible impact on the considered performance metric.

Due to their inner differences, we strongly support the idea that CCN and CDN should complement each other rather than being perceived as two antagonist models. In particular, our numerical results give evidence that CCN should be preferred whenever the content popularity dynamics evolves very quickly, while in the other cases CDN leads to the lower overall traffic exchanged in the network. The relative performance gain depends on the characteristics of the specific instance that was taken into account, and is clearly a function of the set of parameters considered to characterize the network scenario. Finally, we recognize that the simplicity of CCN should reduce the management costs with respect to CDN, but such type of analysis is out of the scope of our work.

3. Content Popularity Evolution Model

This section discusses the content popularity evolution model we used to generate synthetic traffic demand traces. Many research papers (e.g., [24, 25, 26]) agree in supporting the idea that the content popularity dynamics is highly non-stationary, and characterized by a bursty and oscillatory behavior, mostly

[1] CDNs can operate both in Push or Pull mode. Our model formulation considers the best performance bound for the CDN: we let the model select the best between the Push and Pull model, and we do not force the network to behave according to one of these operation modes.

Algorithm 1: Popularity Evolution, Rank-shift Model

Input : $r_o^t, \mathcal{O}, \rho$
Output: r_o^{t+1}

```
for o ∈ O do
    if UniformRandom(0,1) ≤ ρ then
        r' ⇐ UniformDiscreteRandom(1, r_o^t);
        for o' ∈ O do
            if r' ≤ r_o'^t ≤ r_o^t then
                r_o'^{t+1} ⇐ r_o'^t + 1;
            end
        end
        r_o^{t+1} ⇐ r';
    end
end
```

governed by exogenous events. In this subsection we will accurately describe the synthetic traffic model, based on the proposal of Ratkiewicz et al. [26], that we use to generate the input traffic demand mimicking non-linear popularity shifts for a fixed-size object catalog denoted with $\mathcal{O}$. Despite the fact that content churn[2] may increase the realism, it is a common assumption made in the CDN and CCN literature to consider a fixed-size content catalog [27, 28].

Time is discretized into a finite set of time-slots, denoted with $\mathcal{T}$. Each object $o \in \mathcal{O}$ has a time-dependent rank parameter r_o^t which describes how likely that object will be requested during $t \in \mathcal{T}$. The lower the rank of an object, the higher the likelihood that such content will receive requests in that time-slot. We assume all the time slots $t \in \mathcal{T}$ last for the same (and constant) amount of time and such that the popularity rank of every object in t does not change.

Given the object rank r_o^t, the Zipf discrete distribution is often used in the literature to represent the popularity of Internet contents, since it was shown that it is an adequate model for it [9, 28, 29]. The Zipf distribution is charac-

[2]The churn is a measure of the number of individuals moving in or out a given collective over a specific period of time.

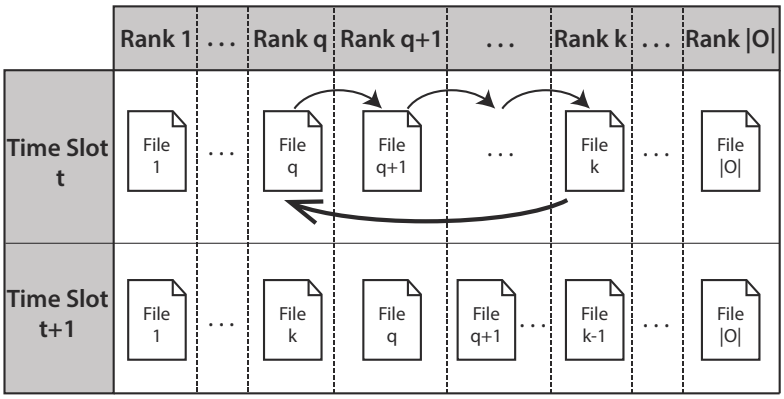
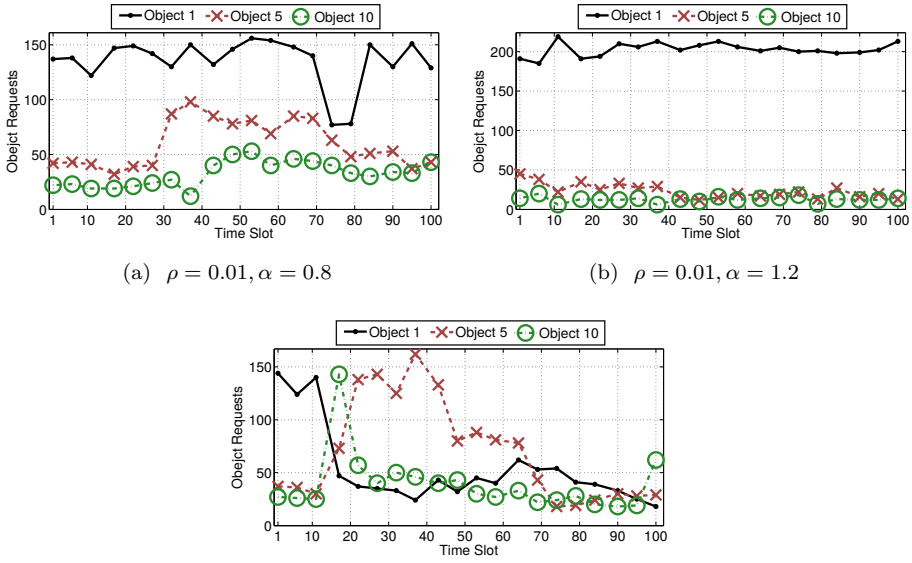


Figure 3: *The Rank-Shift Model.* In this example, at time slot $t+1$, the popularity of the k-th ranked file is updated to the value of q (randomly chosen), thus all the files from q to $k-1$ are shifted of one rank.

terized by the popularity exponent α: the higher the α, the more skewed the requests are.

The time-dependent popularity dynamics is governed by the rank evolution parameter ρ in the 0 to 1 range. At each time slot, the ranking of a given object might evolve to become more popular in the immediate future. We control the speed at which this change happens through the usage of ρ: the higher the ρ value, the faster the popularity evolves. In the extreme cases, by setting $\rho = 0$ we neglect popularity evolution, whereas $\rho = 1$ removes temporal correlation of the requests. When the rank r_o^t of an object is updated, it makes the other objects' rank be shifted to a new (less popular) value.

Algorithm 1 illustrates the pseudocode used to compute the future object rank, given its current value. For each object (Step 1), with probability ρ (Step 2), the algorithm randomly selects a lower popularity class r' (Step 3), making the object suddenly become more popular. In Step. 4, the rank of the other objects is instead shifted to a reduced popularity level, in order to make sure that the popularity rank will be unique among all the objects. An example of object popularity evolution is shown in Fig. 3, where at time slot $t+1$, the popularity of the k-th ranked file is updated to the new value of q, randomly



(a) $\rho = 0.01, \alpha = 0.8$ (b) $\rho = 0.01, \alpha = 1.2$

(c) $\rho = 0.1, \alpha = 0.8$

Figure 4: *Time-Dependent Object Request Evolution*. The figures plot the time-dependent object request evolution as a function of the Zipf α exponent and the rank evolution probability ρ, considering 100 time slots and 10 objects. The figures show the evolution of requests for the objects that in the first time slot have rank 1, 5 and 10, being 1 and 10 the most and least popular ranks, respectively.

chosen.

As illustrated in Figures 4a-4c, by changing the ρ and α parameters, we can easily mimic very different behaviors: a range of different content requests can be generated with such model, thus allowing us to well represent almost any type of request generation process. In particular, by decreasing the α value, content requests are less polarized towards the most popular objects, while higher ρ values make the content popularity dynamics evolve faster.

4. Network Models for Content Distribution

This section discusses the proposed optimization models that we use to study the performance of the CCN and CDN paradigms. Sec. 4.1 presents the *Object Routing model* (OR) with *time-varying* demands. In Sec. 4.2 and Sec. 4.3, we

Table 2: Summary of the Network Optimization Models we propose in this paper.

Model Name	**Caching Strategy**	**Time Horizon**	**Network Type**	**Section**
Object Routing (OR)	-	Many Slots	TCP/IP	4.1
Object Allocation and Routing (OAR)	Optimal	Many Slots	CCN	4.2
OAR - Single Time Slot Heuristic (OAR-TS)	Optimal LFU Random	Single Slot	CCN	4.2
OAR - Single Relocation Time CDN Heuristic (OAR-TR-CDN)	Optimal	Single Slot	CDN	4.3

tailor the OR model to better represent relevant characteristics of CCN and CDN, respectively.

4.1. Object Routing Model with Time-Varying Demands

In this subsection we describe our proposed *Object Routing model* (OR) with *time-varying* demands. Such model well describes a TCP/IP network that does not have any caching functionality. In the following subsections we will further extend the OR model, taking into account the features that characterize CCN and CDN architectures. As summarized in Table 2, three extensions of the OR model will be presented:

1. *Object Allocation and Routing (OAR)*, a model tailored for CCN, that finds the optimal solution on the overall time horizon;
2. *OAR - Single Time Slot Heuristic (OAR-TS)*, tailored for CCN, which chooses the optimal solution for single time slots;
3. *OAR - Single Relocation Time CDN Heuristic (OAR-TR-CDN)*, which is the equivalent for CDN of OAR-TS.

We model the network as an directed graph $G(V, A)$, where V is the set of nodes and A the set of arcs. The set of nodes V is partitioned into three disjoint sub-sets: *consumers* (denoted by $\mathcal{C}$), *producers* (denoted by $\mathcal{P}$) and *routers* (denoted by $\mathcal{R}$), such that $V = \mathcal{C} \cup \mathcal{P} \cup \mathcal{R}$. Furthermore, we denote with $\mathcal{T}$ the set of time slots, whereas $\mathcal{O}$ represents the set of objects that can be retrieved from the network, also known as the *catalog*. It is important to mention that, as frequently assumed in the network planning literature, both the *consumer* and *producer* nodes must be interpreted as *test points* at which an aggregate of traffic demands can either be requested or served.

Forward and backward arcs of node $i \in V$ are denoted with $FS(i)$ and $BS(i)$, respectively. We denote with d_c^{to} the demand of consumer $c \in \mathcal{C}$ for object $o \in \mathcal{O}$ at time $t \in \mathcal{T}$. The demand is expressed in data size units (e.g., Mbytes); moreover, as in the literature [18], we assume that traffic requests are *inelastic*, meaning that they cannot be postponed to subsequent time slots, but they must be served within the end of the current time slot. In order to satisfy the demands, producers can serve the subset of objects they own. For each producer $p \in \mathcal{P}$ and object $o \in \mathcal{O}$, we denote with a_p^o the producer-object allocation, where:

$$a_p^o = \begin{cases} 1, & \text{if object } o \text{ is available at producer } p \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

All the routers in the network perform routing and forwarding. We denote the link capacity of arc $e \in A$ with b_e, while y_e^{to} denotes the time-dependent flow variable which represents the amount of data that arc $e \in A$ is delivering for object $o \in \mathcal{O}$ in time slot $t \in \mathcal{T}$. For the sake of clarity, Table 3 summarizes the notation we will use in our optimization models.

We begin by describing the *Object Routing* model we use to compute the optimal packet routing to minimize the overall network traffic. By solving this model, we determine the performance bound of a network that does not support any type of caching functionality. In the following sections (viz., Sec. 4.2 and 4.3) we will extend the OR model in order to describe the behavior of a CCN and a CDN architecture, respectively. Given the above definitions and assump-

Table 3: Summary of the notation used in our optimization models

Parameters	
$G = (V, A)$	Directed graph G, composed by nodes V and arcs A.
$\mathcal{C}$	Set of Consumers test points
$\mathcal{P}$	Set of Producers test points
$\mathcal{R}$	Set of Routers
$\mathcal{O}$	Set of Objects
$\mathcal{T}$	Set of Time Slots
$\mathcal{D}$	Set of CDN Nodes test points
d_c^{to}	Traffic demand generated by consumer c for object o in time slot t
b_e	Link capacity of arc $e \in A$
$FS(i)$	Set of forward arcs of node $i \in V$
$BS(i)$	Set of backward arcs of node $i \in V$
a_p^o	0-1 parameter to indicate if producer p is publishing object o
S	Maximum number of objects that each router can cache
Q	A large number
N	Maximum number of CDN nodes that can be deployed
m	Maximum memory that a CDN node can use for caching
M	Total memory shared by all CDN nodes for caching

Decision Variables	
x_r^{to}	0-1 variable indicating if router r is caching object o at time t
y_e^{to}	Data flow of object o on arc $e \in A$, during time slot t

tions, we formulate the optimal *Object Routing* model (OR) with *time-varying* demands as follows:

$$\text{minimize} \sum_{\substack{\forall o \in \mathcal{O} \\ \forall t \in \mathcal{T} \\ \forall e \in A}} y_e^{to} \tag{2}$$

subject to:

$$\sum_{\forall e \in FS(r)} y_e^{to} = \sum_{\forall e \in BS(r)} y_e^{to} \qquad \forall (r,o,t) \in \mathcal{R} \times \mathcal{O} \times \mathcal{T} \tag{3}$$

$$\sum_{\forall o \in \mathcal{O}} y_e^{to} \leq b_e \qquad \forall e \in A \times \mathcal{T} \tag{4}$$

$$\sum_{\forall e \in BS(c)} y_e^{to} = d_c^{to} \qquad \forall (c,o,t) \in \mathcal{C} \times \mathcal{O} \times \mathcal{T} \tag{5}$$

$$y_e^{to} \leq b_e \cdot a_p^o \qquad \forall (p,o,t) \in \mathcal{P} \times \mathcal{O} \times \mathcal{T}, e \in FS(p) \tag{6}$$

$$y_e^{to} \geq 0 \qquad \forall (e,o,t) \in A \times \mathcal{O} \times \mathcal{T}. \tag{7}$$

The objective function (2) minimizes the total traffic transferred across all network links.

The set of constraints (3) imposes the flow balance condition at each router. Constraints (4), bound the total link capacity that can be used on each arc. The set of constraints (5) makes sure that the network satisfies, in each time slot, all consumers' demand. Producers can only serve the subset of objects they possess, and this condition is expressed by the set of constraints (6). Lastly, constraints (7) impose that router-router, producer-router and router-consumer flows are non-negative.

As represented in the OR model, the behavior of the network in each time slot is such that it does not depend on the solution obtained in other time slots. Therefore, it is possible to speed-up the model resolution by solving the routing problem independently in each time slot, and then aggregating the solution to compute the final objective function value.

4.2. Model for Content-Centric Network

In this section, we present the *Object Allocation and Routing* (OAR) model, an extension of the OR model tailored for a CCN.

The OAR model extends the OR model adding *caching capabilities* to the routers. Given router $r \in \mathcal{R}$, object $o \in \mathcal{O}$ and time slot $t \in \mathcal{T}$, the router cache state is denoted by the binary variable x_r^{to}, which is such that:

$$x_r^{to} = \begin{cases} 1, & \text{if object } o \text{ is cached at router } r \text{ at time slot } t \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

In particular, we model a network where the cache is uniformly spread among all the nodes. Our model solves the *joint* optimal Object Allocation and Routing (OAR) problem, where the consumers express a *time-varying demand*. We can therefore formulate the mixed integer linear programming (MILP) model for OAR as follows:

$$\text{minimize} \sum_{\substack{\forall o \in \mathcal{O} \\ \forall t \in \mathcal{T} \\ \forall e \in A}} y_e^{to} \tag{9}$$

subject to constraints (4)-(7), and:

$$\sum_{e \in FS(r)} y_e^{to} \leq Q \cdot x_r^{to} + \sum_{e \in BS(r)} y_e^{to} \qquad \forall (r,o,t) \in \mathcal{R} \times \mathcal{O} \times \mathcal{T} \tag{10}$$

$$x_r^{t,o} \leq x_r^{t-1,o} + \sum_{\forall e \in BS(r)} y_e^{t-1,o} \qquad \forall (r,o,t) \in \mathcal{R} \times \mathcal{O} \times (\mathcal{T} \setminus \{0\}) \tag{11}$$

$$\sum_{\forall o \in \mathcal{O}} x_r^{to} \leq S \qquad \forall (r,t) \in \mathcal{R} \times \mathcal{T} \tag{12}$$

$$x_r^{0,o} = 0 \qquad \forall (r,o) \in \mathcal{R} \times \mathcal{O} \tag{13}$$

$$x_r^{to} \in \{0,1\} \qquad \forall (r,o,t) \in \mathcal{R} \times \mathcal{O} \times \mathcal{T}. \tag{14}$$

The objective function (9) is the same as the one proposed for the OR model. The set of constraints (10) imposes flow-balance at each router, by also taking into account its caching capabilities. In particular, Q is a large number such that, when router r_1 is caching object o at time t (that is, $x_{r_1}^{t,o} = 1$), r_1 can directly serve all the incoming requests for that object[3]. Link capacity

[3]We set $Q = \max_{\forall o \in O, t \in T} \sum_{\forall c \in C} d_c^{to}$, even if it is also possible to use larger values for which the same final results will be obtained.

constraints imposed by (4) are still valid in OAR, even when the router behaves as a cache.

In CCN, each node acts independently from all the others by choosing which content it should store according to the *local* information available. This means that each node can choose to store a content in its local cache if and only if in the previous time slot it has forwarded such object, or if it was already caching such data. This constraint is imposed by (11) on all the caching routers in the network.

In (12), we make sure that each caching router stores at most S objects in its local cache. In (13), we make sure that the caches are empty at the initial time slot (in a sort of initialization step); finally, the set of constraints (14) forces the variable x_r^{to} to be binary. It is important to note that the OAR model supports multipath packet routing, a native feature provided by CCN, and it also supports *off-path* caching. In fact, each router chooses the face on which packets should be forwarded knowing also the availability of content replicas in the caching storage of all the other nodes. Supporting off-path caching is in line with our goal of computing the performance bound of the CCN network.

Due to the fact that CCN routers can cache only the subset of objects they have seen in the immediate past, current network behavior strongly influences possible future choices, making every time slot be linked to all the others, as modeled by constraints (11). However, finding the optimal solution over the complete set of time slots is computationally very expensive, since the model has to consider all the demands on a global scale. Moreover, this optimization strategy assumes that the model can perform the optimal choice by knowing also *"future"* demands.

As shown by numerical results presented in Sec. 5.1, knowing the future (as OAR does) seems to provide only negligible improvements to the objective function, when compared to the alternative case where the solver knows the current demands and the previous state of the system. We call this latter model OAR *Single Time Slot Heuristic* (OAR-TS). OAR-TS lets us find a close to optimal solution to OAR, saving significant amount of time for the computation.

The OAR-TS model is illustrated hereafter:

$$\text{minimize} \sum_{\substack{\forall o \in \mathcal{O} \\ \forall e \in A}} y_e^o \tag{15}$$

subject to:

$$\sum_{e \in FS(r)} y_e^o \leq Q \cdot x_r^o + \sum_{e \in BS(r)} y_e^o \qquad \forall (r_1, o) \in \mathcal{R} \times \mathcal{O} \tag{16}$$

$$\sum_{\forall o \in \mathcal{O}} y_e^o \leq b_e \qquad \forall e \in A \tag{17}$$

$$\sum_{\forall e \in BS(c)} y_e^o = d_c \qquad \forall (c, o) \in \mathcal{C} \times \mathcal{O} \tag{18}$$

$$y_e^o \leq b_e \cdot a_p^o \qquad \forall (p, o) \in \mathcal{P} \times \mathcal{O}, e \in FS(p) \tag{19}$$

$$\sum_{\forall o \in \mathcal{O}} x_r^o \leq S \qquad \forall r \in \mathcal{R} \tag{20}$$

$$x_r^o \leq K_r^o \qquad \forall (o, r) \in \mathcal{O} \times \mathcal{R} \tag{21}$$

$$y_e^o \geq 0 \qquad \forall (e, o) \in A \times \mathcal{O} \tag{22}$$

$$x_r^o \in \{0, 1\} \qquad \forall (r, o) \in \mathcal{R} \times \mathcal{O}. \tag{23}$$

The objective function (15) as well as constraints (16)-(20) and (22)-(23) can easily be derived from the corresponding constraints of the OAR formulation, by removing the time-slot index.

We model time-slots relations in constraints (21), where in each time slot $t \in \mathcal{T}$ we use the binary parameter K_r^o to impose restrictions on the subset of objects that a given router $r \in \mathcal{R}$ can cache. In particular, we emulate the original behavior studied in the OAR formulation by setting $K_r^o = 0$ for the initial slot $t = 1$, whereas for the other time slots $t > 1$, we set $K_r^o = 1$ if, in $t - 1$, $x_r^o = 1 \vee \sum_{\forall e \in BS(r)} y_e^o > 1$, otherwise we set $K_r^o = 0$. Such condition checks whether in the previous time slot the router was already caching object o, or it forwarded at least one traffic unit for it.

By using other strategies to set the K_r^o parameter we can find tighter performance bounds for *real caching policies*; in particular we can emulate the "*Least Frequently Used*" (LFU) policy, forcing caches to store the objects that are re-

quested more frequently, as well as the "Random" caching policy, making caches store the objects randomly [30].

More in depth, we emulate the LFU policy by computing Ω_t^{ro}, the cumulative traffic that router $r \in \mathcal{R}$ has forwarded for a given object $o \in \mathcal{O}$ up to time slot $t \in \mathcal{T}$, which is defined as: $\Omega_t^{ro} = \Omega_{t-1}^{ro} + \sum_{e \in FS(r)} y_e^o$. For each router, we sort the Ω_t^{ro} values in decreasing order; we then remove all those values referred to objects $o \in \mathcal{O}$ that the router has not seen in the previous time slot $t-1$, i.e. all those objects such that the condition $x_r^o = 1 \vee \sum_{\forall e \in BS(r)} y_e^o > 1$ does not hold. Let Ω_{St}^r be the S-th element of such sorted list. We set $K_r^o = 1$ if $\Omega_t^{ro} \geq \Omega_{St}^r$, otherwise it is set to zero. As a result, in $t+1$ the S most frequently requested objects are stored in the cache of the r-th router, according to its local cumulative traffic data.

The random cache policy can be implemented using even an easier algorithm: we randomly sample S objects such that for each of them, $o \in \mathcal{O}$, the following condition holds: $x_r^o = 1 \vee \sum_{\forall e \in BS(r)} y_e^o > 1$.

Computing instead a better optimization model for the LRU policy is quite involving: since we aggregate all the traffic in the entire time slot we lose information on the sequence of requests received by the routers, and for such motivation we do not attempt to model this caching policy, which is instead studied in our simulation analysis.

4.3. Model for Content-Delivery Network

In this section, we illustrate the model used to find the performance bound of a CDN architecture. Since we are mostly interested in studying the performance of the network *after* the physical deployment of the nodes, we assume that the replica server placement problem has already been solved *a-priori*. As discussed in the related work section (6.1), many proposals to solve the placement problem have already been formulated in the literature, and among all of them we use the *k-median* model [11] since it is a simple strategy that only depends on the nodes distribution in the network.

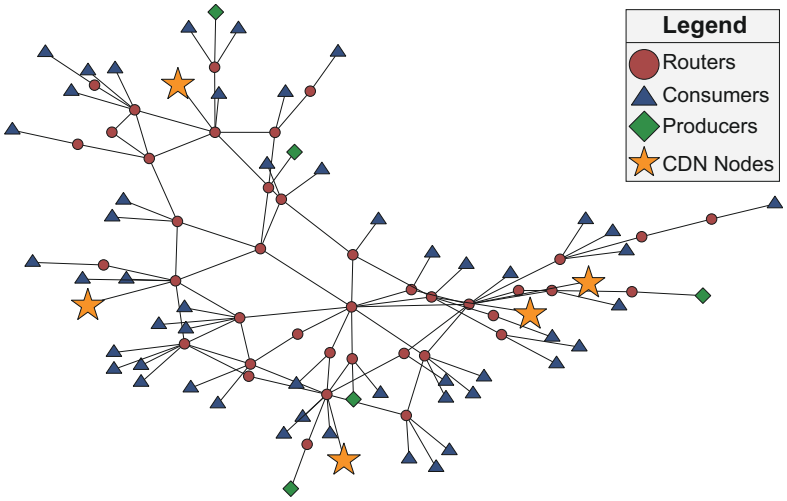


Figure 5: *K-median model*. Example illustrating the placement of 5 CDN replica servers according to the solution of the *k-median model*, in the Géant network topology, with uniform distribution of 50 consumers and 5 producers.

An example showing how the *k-median* model distributes the CDN nodes in the Géant network topology [31] is shown in Fig. 5. In such example, 50 consumers and 5 producers are spread uniformly in the network that is composed of 40 routers, while the *k-median* solution distributes 5 CDN nodes placing them in positions close to the consumers' locations.

Once that we have found the optimal CDN replica server allocation, we can then solve the *joint object placement and request routing problem*, as we did in the previous section with the OAR and the OAR-TS model. Two important features that our optimization model takes into account are:

(1) Each CDN node can cache whatever content available. In fact, the CDN owner can optimize such allocation pushing any content on any replica server.

(2) Since this content migration is quite costly, the objects stored in a CDN node evolve with slower frequency than the one that we can get for CCN.

It is straightforward to address requirement (1), since we can simply relax a

constraint in the OAR-TS model formulation. On the other hand, in order to enforce requirement (2), we introduce another concept that we call "*Relocation Time*", $\mathcal{T}_r$. The relocation time is a subset of consecutive time-slots $\mathcal{T}_r \subseteq \mathcal{T}$ during which the content cached in each CDN node is "frozen" and cannot be changed.

The set of CDN nodes is denoted with $\mathcal{D}$. We then formulate the *Optimal Allocation and Routing, Single Relocation Time Heuristic* model for CDN (OAR-TR-CDN) as follows:

$$\text{minimize} \sum_{\substack{\forall o \in \mathcal{O} \\ \forall e \in A}} y_e^o \tag{24}$$

subject to:

$$\sum_{\forall e \in FS(r)} y_e^o = \sum_{\forall e \in BS(r)} y_e^o \qquad \forall (r,o) \in \mathcal{R} \times \mathcal{O} \tag{25}$$

$$\sum_{\forall o \in \mathcal{O}} y_e^o \leq b_e \cdot |\mathcal{T}_r| \qquad \forall e \in A \tag{26}$$

$$\sum_{\forall e \in BS(c)} y_e^o = \sum_{\forall t \in \mathcal{T}_r} d_c^{to} \qquad \forall (c,o) \in \mathcal{C} \times \mathcal{O} \tag{27}$$

$$y_e^o \leq a_p^o \cdot b_e \cdot |\mathcal{T}_r| \qquad \forall (p,o) \in \mathcal{P} \times \mathcal{O}, e \in FS(p) \tag{28}$$

$$\sum_{\forall o \in \mathcal{O}} x_d^o \leq S' \qquad \forall d \in \mathcal{D} \tag{29}$$

$$y_e^o \leq b_e \cdot |\mathcal{T}_r| \cdot x_d^o \qquad \forall (d,o) \in \mathcal{D} \times \mathcal{O}, e \in FS(d) \tag{30}$$

$$y_e^o \geq 0 \qquad \forall (e,o) \in A \times \mathcal{O} \tag{31}$$

$$x_r^o \in \{0,1\} \qquad \forall (r,o) \in \mathcal{R} \times \mathcal{O}. \tag{32}$$

The OAR-TR-CDN objective function (24) is similar to the one proposed for the other models, since we want to minimize the overall network traffic.

In (25) we set the flow balance constraints, while in (26) we set the capacity constraints. The overall consumer demand should be satisfied, as enforced by constraints (27), where we aggregate consumers' demands on all the time-slots in the relocation time $t \in \mathcal{T}_r$. In (28) we force producers to offer the subset of objects they own.

The set of constraints (29) limits the available caching space at each CDN router, while in (30) we limit the subset of objects that a CDN node can serve to those that it is caching. Finally, non-negativity constraints for flow variables are imposed in (31), while in (32), we make sure that the x_r^o variable is binary.

The relocation time is a subset of consecutive time slots, and is used in the OAR-TR-CDN model to aggregate the entire demand in one *virtual* slot. For this reason in constraints (26), (28) and (30) we multiply the capacity by $|\mathcal{T}_r|$, a non-dimensional quantity that is the number of time-slots considered in the given relocation time. Such choice permits to reduce the computational time required to solve the optimization problem.

Comments

The OAR-TS and OAR-TR-CDN models capture relevant characteristics of the CCN and CDN architectures, respectively. In particular, while in OAR-TS all the routers are equipped with caching storage, in OAR-TR-CDN only the subset of CDN nodes implement caching functionalities. Moreover, OAR-TS sets a constraint on the objects that each router can cache, restricting such subset to all the data that the router has forwarded in the previous time slot. On the other hand, CDN nodes can cache whatever content they want, and we use the relocation time to control the speed at which the CDN reacts to popularity evolution. Furthermore, our analysis focuses on the performance bounds of CCN and CDN: in particular, we are not considering management cost savings that the CCN architecture can obtain with respect to CDN.

Our proposed OAR-TR-CDN model can further be extended to explicitly take into account the cost for object migration. For the sake of simplicity, we focus on the scenario where CDN nodes retrieve the objects from the closest producer publishing them, even though this condition can easily be relaxed. Hereafter we report the extended version of such optimization model, while in the numerical results section we show that the cost impact of this object movement is negligible.

Let $z_{d,e}^o$ be an input parameter that represents the amount of data that

should be sent through network links if CDN node d retrieves object o from the closest producer publishing such object. Let $x_d^{t-1,o}$ be a 0-1 input parameter that represents the state of the cache for the CDN nodes in the previous relocation time $t-1$, and let δ_d^o be a variable that is set to 1 if object o should be moved to the cache of CDN node d in the current time slot. To take into account object movement costs in the CDN model formulation, we replace the objective function (24) with the one provided below:

$$\text{minimize} \sum_{\substack{\forall o \in \mathcal{O} \\ e \in A}} y_e^o + \sum_{\substack{\forall o \in \mathcal{O} \\ \forall d \in \mathcal{D} \\ \forall e \in A}} \delta_d^o z_{d,e}^o. \tag{33}$$

We add the following set of constraints:

$$\delta_d^o \geq x_d^o - x_d^{t-1,o} \qquad \forall (d,o) \in \mathcal{D} \times \mathcal{O} \tag{34}$$

$$\delta_d^o \in \mathbb{R}^+ \qquad \forall (d,o) \in \mathcal{D} \times \mathcal{O}. \tag{35}$$

We replace constraints (26) with those provided below:

$$\sum_{\forall o \in \mathcal{O}} \left(y_e^o + \sum_{\forall d \in \mathcal{D}} \delta_d^o \, z_{d,e}^o \right) \leq b_e \cdot |\mathcal{T}_r| \qquad \forall e \in A. \tag{36}$$

In particular, with these modifications our formulation optimizes the CDN object placement taking into account the cost to move the object to the given CDN node.

5. Numerical Results

This section presents the results obtained formulating the proposed optimization models in OPL and solving them using the CPLEX solver (version 12.6.1) [32], and running on an Intel i7-3770 CPU @ 3.40GHz with 16 GB of RAM.

In Sec. 5.1 we compare the objective function obtained using OAR with the heuristic OAR-TS solution. Sec. 5.2 extensively presents and discusses the numerical results we recorded while comparing the performance bounds of CCN and CDN. The results concerning the computation time spent to find the optimal solution in the different formulations are illustrated in Sec. 5.3. In Sec. 5.4 we

study the effect of the approximations we adopted, whereas in Sec. 5.5 we present the results concerning the impact of each of the characterizing assumptions we made to differentiate CCN with respect to CDN. Finally, the results obtained in the simulations are discussed in Sec. 5.6.

5.1. Comparison of OAR and OAR-TS

In this subsection we compare the results obtained using OAR and OAR-TS and show that, in the considered scenarios, the bounds of the two models are very close to each other, even though OAR is computationally more demanding than OAR-TS. Unless stated otherwise, Table 4 summarizes the parameters used to perform the analysis.

The topology we consider for this analysis is the Netrail topology with 10^7 objects partitioned into 100 popularity classes (as in [33]). We compute the optimal solution considering 100 time slots, while we attach consumers and producers to 10 and 5 test points, respectively. We run the OAR model limiting the overall running time to 10 hours and observing a final MIP gap always below 2% of the optimum solution. Traffic demands are generated setting $\alpha = 0.8, \rho = 0.2$. For the same scenario, CPLEX can always find the optimal solution of the OAR-TS model in less than ten minutes for the whole time horizon.

As portrayed in Fig. 6, the OAR and OAR-TS curves are practically overlapping, in fact the gap between the two is always below 1%. This behavior clearly shows that knowing future demands (as OAR does) can lead only to negligible improvements in the objective function, and therefore it legitimates the use of our proposed OAR-TS heuristic to derive a close to optimal solution.

Due to the very high memory usage required by CPLEX to solve OAR, unfortunately we could not take into account larger topologies to compare it with OAR-TS. Thanks to its modest size, Netrail is the only topology on which we can perform one such analysis. However, the very narrow approximation introduced by OAR-TS with respect to OAR is a promising result that confirms that knowing future traffic requests may only slightly improve the final solution, and is therefore negligible.

Table 4: Parameters of the Numerical Results

Numerical Results: Common Parameters	
Topologies	Netrail (7 nodes), Abilene (11 nodes) Sprint (11 nodes), Claranet (15 nodes) Airtel (16 nodes), Géant (40 nodes)
Number of Consumers	10 000, connected at 10 test-points
Number of Producers	5 000, connected at 5 test-points
Zipf α Exponent	$\{0.8, 1.2\}$
Content Popularity Evolution ρ	In the range from 10^{-4} to 0.99
Link Rate	50 000 objects per time slot
Consumer Demand	10 objects per time slot, per consumer
Cache Size per Router	In the range 1-5% of the total catalog
Catalog Size	10^7 objects - 100 popularity classes

Numerical Results: Optimization Models	
Number of Time Slots	100
CDN Relocation Time	In the range 1-15, default value: 3
Number of CDN nodes	In the range 1-10, default value: 3

Numerical Results: Simulations	
Cache Replacement Policies	Random (RND), Least frequently used (LFU), Least recently used (LRU)
Time Slot Size	1 second
Trace Length	1 hour

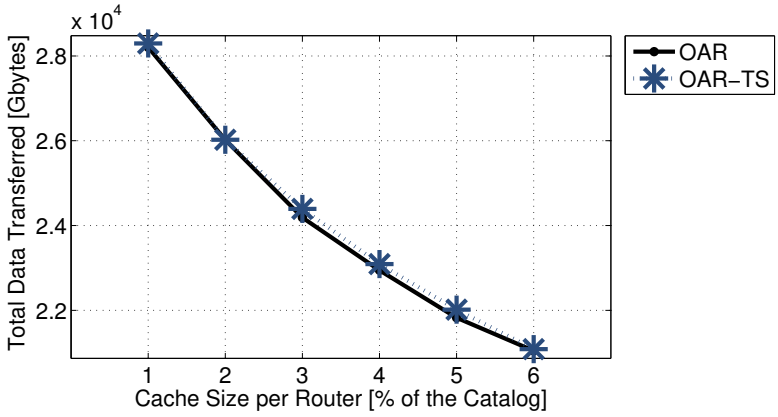
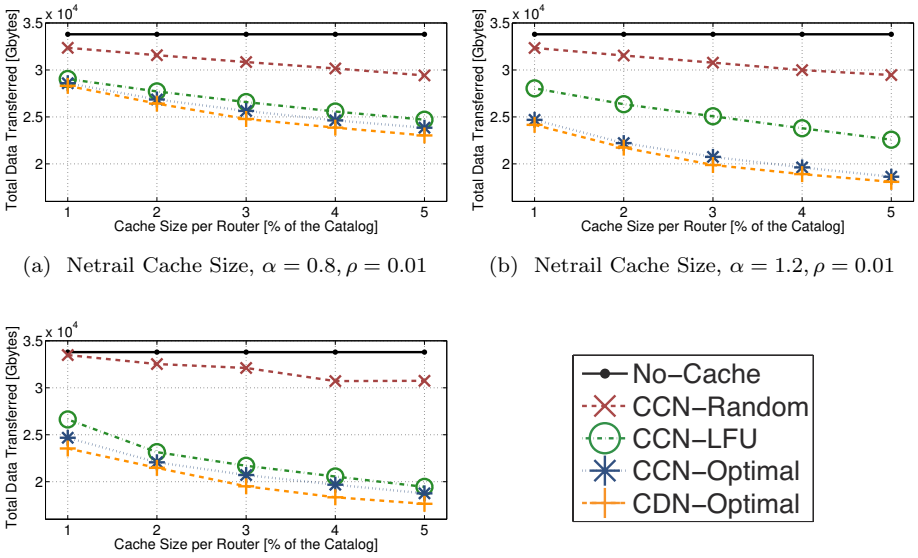


Figure 6: *Comparison of OAR and OAR-TS.* We solve the Netrail topology with a catalog of 100 object classes and 100 time slots, with $\alpha = 0.8, \rho = 0.2$. The gap between the solution of OAR and OAR-TS is negligible (less than 1%).

5.2. Performance Comparison of CCN and CDN Using Optimization Models

In this section we present the results obtained performing an evaluation of the optimization models we designed. As shown in Table 4, we considered 6 different network topologies whose size is in line with [10, 30, 34, 35]. We provide the full set of plots and the code used to generate the numerical results at [36], while, for the sake of brevity, hereafter we present the results obtained with Netrail and Géant, since the trends observed for the other topologies are in between the values observed for them.

As shown in Table 4, in all the scenarios we consider 10 000 consumers and 5 000 producers connected respectively to 10 and 5 test points spread uniformly in the network, in line with [35] where the authors consider 5 test points for the producers. The content catalog we take into account is composed of 10^7 objects partitioned into 100 popularity classes, where the average object size is 1 Mbyte, as done in [27, 37, 33]. Each consumer requests 10 objects in each time slot, in agreement with [38]. Object requests are distributed according to the rank-shift model discussed in Sec. 3, and we consider the ρ exponent in the 10^{-4} to 0.99 range, while the alpha exponent of the Zipf distribution can either be $\alpha = 0.8$ or $\alpha = 1.2$, as done in [39] and in line with [9, 10]. We consider different



(a) Netrail Cache Size, $\alpha = 0.8, \rho = 0.01$

(b) Netrail Cache Size, $\alpha = 1.2, \rho = 0.01$

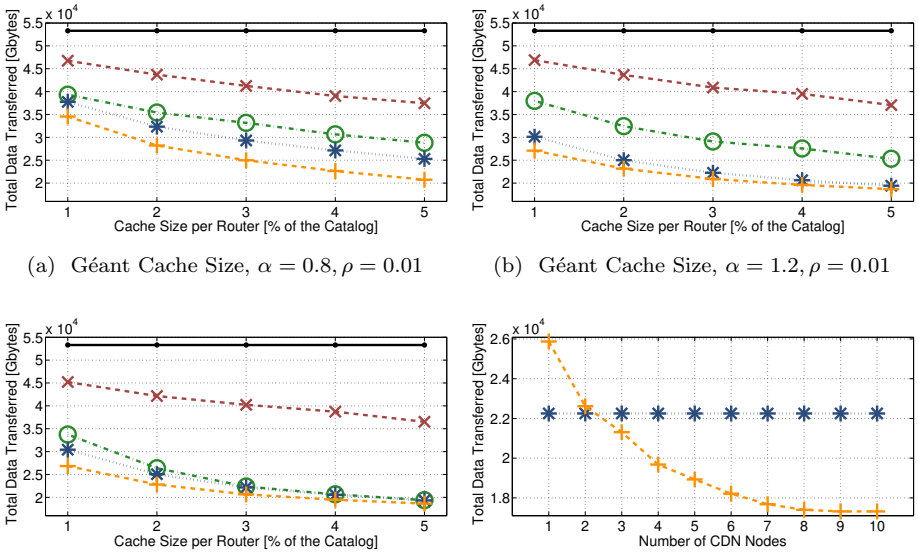
(c) Netrail Cache Size, $\alpha = 1.2, \rho = 10^{-4}$

(d) Legend

Figure 7: *Netrail Topology, Effect of the Cache Size.* Figures 7a-7c show the effect of the cache size in the Netrail topology, for different values of the Zipf α exponent, as well as the popularity evolution probability ρ. The legend of Fig. 7d is common to all the plots in Fig. 7-9.

cache sizes, such that they can store from 1 to 5% of the total number of objects available, as considered in other studies (e.g., [9, 27]). The number of test points for the CDN nodes we deploy is up to 10, whereas we consider a relocation time between 1 and 15, meaning that in our analysis the CDN might be "as fast as" a CCN or up to 15 times slower than that. If not stated otherwise, 3 CDN nodes will be deployed and they will have a relocation time equal to 3. We believe that these values can be used to perform a *fair* comparison of CCN and CDN: by deploying such a small number of CDN nodes we do not bias the analysis in favor of this latter architecture. The performance metric we consider is the total network traffic, since it is the objective function we take into account for the optimization models.

The behavior of the Netrail topology as a function of the cache size is shown in Fig. 7a (for $\alpha = 0.8, \rho = 0.01$), Fig. 7b (for $\alpha = 1.2, \rho = 0.01$) and Fig. 7c (for $\alpha = 1.2, \rho = 10^{-4}$). Considering the *No-Cache* curves for different combinations



(a) Géant Cache Size, $\alpha = 0.8, \rho = 0.01$

(b) Géant Cache Size, $\alpha = 1.2, \rho = 0.01$

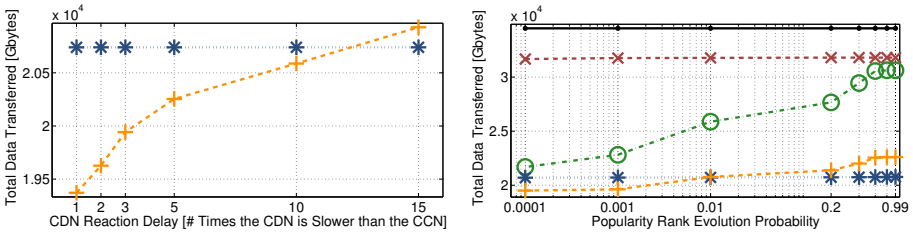
(c) Géant Cache Size, $\alpha = 1.2, \rho = 10^{-4}$

(d) Géant CDN Nodes, $\alpha = 1.2, \rho = 0.01$

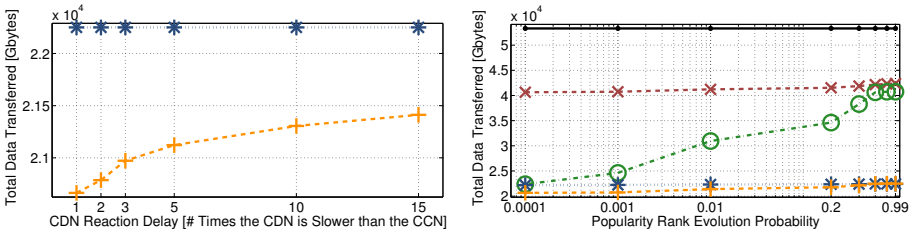
Figure 8: *Géant Topology, Effect of the Cache Size.* Figures 8a-8c show the effect of the cache size in the Géant topology, for different values of the Zipf α exponent, as well as the popularity evolution probability ρ. Figure 8d shows the effect of the number of CDN nodes in Géant.

of α and ρ values (as in Fig. 7a-7c), we observe that a network that does not have caching functionalities leads to the same overall traffic (about 3.34 Tbytes for Netrail), even for different popularity evolution parameters. By introducing caching functionalities, the total network traffic can be reduced significantly: in the Netrail topology, caching reduces traffic up to 46% (Fig. 7c, CDN-Optimal caching policy, 5% cache size), while in the Géant topology, traffic savings can reach 66% (Fig. 8c, CDN-Optimal caching policy, 5% cache size).

An interesting observation on the random caching policy is that it leads to a total traffic that is rather independent with respect to the popularity evolution parameters α and ρ: the total traffic for CCN-Random is on average 8% lower than for No-Cache. On the contrary, the LFU caching policy is very sensitive to the value of ρ: the lower the speed at which the popularity evolves, the better the objective function is, as shown in Fig. 7b and 7c. In particular, while in



(a) Netrail Relocation Time, $\alpha = 1.2, \rho = 0.01$ (b) Netrail Evolution Probability, $\alpha = 1.2$



(c) Géant Relocation Time, $\alpha = 1.2, \rho = 0.01$ (d) Géant Evolution Probability, $\alpha = 1.2$

Figure 9: *Effect of the Relocation Time, Effect of the Evolution Probability.* Figures 9a and 9c show the effect of the relocation time in Netrail and Géant topologies, respectively. The effect of the evolution probability on the overall traffic is instead shown in Figures 9b and 9d in Netrail and Géant, respectively.

Fig. 7b LFU scores on average 25% traffic reduction compared to No-Cache, in Fig. 7c the same performance gain raises to 35%.

The topology size has a strong impact on the results, in particular, caching is more beneficial in larger topologies. In fact, if we compare the No-Cache curves of Fig. 7a and 8a, we observe 38% more traffic in Géant than Netrail. However, if we assume that caching functionalities are deployed in the network, as in the CDN-Optimal case, Géant requires on average only 5% more traffic than Netrail when $\alpha = 0.8$, and less than 8% more, when $\alpha = 1.2$ as in Fig. 7a,8a and 7b,8b.

Another remarkable result regards the efficiency of the different distribution architectures for different topologies. As a matter of fact, while in the small Netrail topology CCN and CDN almost exhibit the same performance, in Géant there is a large gap between the two solutions: on average, CDN reduces the total network traffic 14% more than what CCN does, leading to an overall 51%

performance gain (on average), with respect to the total traffic transmitted in the No-Cache scenario, when $\alpha = 0.8$. This behavior is caused by the fact that the optimal solution of the CCN network tends to scatter many copies of the same popular objects in all the caching nodes, whereas cache duplication is significantly reduced by aggregating the storage on few CDN surrogates occupying central locations in the network topology (*k-median* positioning). This key finding is even more evident in Fig. 8d, where we observe that few CDN nodes in the Géant topology (only 3 in the considered scenario) are sufficient to make CDN reach better performance values than those obtained with CCN.

The sensitivity of the objective function to the relocation time $|\mathcal{T}_r|$ is depicted in Fig. 9a and 9c, for the Netrail and Géant topology, respectively. In both cases we deploy 3 CDN nodes. In the Netrail topology (Fig. 9a), even if we assume that CDN nodes are 10 times slower to react to popularity changes than the CCN counterpart, CDN still shows better performance. On the other hand, for the Géant topology (Fig. 9c), CDN always leads to better performance from $|\mathcal{T}_r| = 1$, up to $|\mathcal{T}_r| = 15$.

Also the ρ parameter that drives the speed of the content popularity evolution affects the objective function, as depicted in Fig. 9b for Netrail and Fig. 9d for Géant. The LFU policy is the one that is subject to the largest variation: by increasing the ρ value we reduce the locality of reference of the requests, and this, in turn, penalizes the LFU strategy. It is very interesting to note that in the Netrail topology, when $\rho < 0.01$ CDN is to be preferred, whereas for larger ρ values, CCN leads to the best performance value.

As portrayed in the many plots and discussed throughout this section, we can not claim that either one of the CCN and CDN architectures should always be preferred over the other for its intrinsic superior performance. On the other hand, the value of the objective function strongly depends on the set of parameters used to characterize the specific network scenario.

Table 5: Summary of the Average Execution Time.

Topology	**CCN Nocache**	**CCN OPT**	**CDN**
Abilene	$18\,256 \pm 180$ ms.	$418\,153 \pm 21\,441$ ms.	$57\,739 \pm 3\,416$ ms.
Airtel	$50\,122 \pm 1\,034$ ms.	$306\,532 \pm 15\,301$ ms.	$49\,291 \pm 2\,874$ ms.
Claranet	$21\,528 \pm 652$ ms.	$282\,151 \pm 13\,743$ ms.	$58\,129 \pm 3\,427$ ms.
Géant	$85\,475 \pm 24$ ms.	$2\,477\,916 \pm 179\,243$ ms.	$296\,054 \pm 22\,670$ ms.
Netrail	$11\,226 \pm 189$ ms.	$86\,394 \pm 8\,119$ ms.	$26\,217 \pm 1\,626$ ms.
Sprint	$15\,098 \pm 42$ ms.	$228\,975 \pm 9\,762$ ms.	$48\,433 \pm 2\,847$ ms.

5.3. Computation Time Results

Previous works such as [40, 30] have shown that the optimal content placement problem is NP-Hard.

In Table 5 we summarize the average execution time (in milliseconds) we observed while finding the solutions of the numerical results. In particular we observe that content caching dramatically increases the model's complexity (i.e.: the time necessary to solve CCN Nocache is significantly lower than the one required for CCN OPT and CDN). Moreover, the CDN formulation requires less time than CCN because the presence of the relocation time lets us aggregate the number of time slots, reducing the overall number of iterations required to find the optimal solution compared to CCN.

Finally, we would like to point out that this work focuses on the analysis of the network performance, rather than finding efficient algorithms to solve this problem. As a matter of fact, we do not have any special time constraint that prevented us to perform one such analysis using offline optimization techniques.

5.4. Impact of Approximations

In the previous sections we solved our proposed optimization models assuming that the content catalog was composed of 10^7 different objects partitioned into 100 popularity classes; moreover, while considering the CDN, we did not take into account the extra-cost to move the objects to the CDN surrogates. In this section we evaluate the effect of both these approximations.

Table 6: Impact of Approximations

Effect of Object Quantization - Géant $\lvert\mathcal{D}\rvert = 1, \alpha = 0.8$		
Number of Objects	**Number of Object Classes**	**Average Error**
10^5	100	2.6%
10^6	100	4.0%
10^7	100	4.3%

Effect of CDN Object Movement	
Scenario	**Traffic to Move Objects to the CDN**
Netrail $\rho = 0.99, \alpha = 0.8$	$< 0.001\%$
Géant $\rho = 10^{-4}, \alpha = 0.8$	$< 0.001\%$
Géant $\rho = 0.99, \alpha = 0.8$	$< 0.02\%$

Despite the fact that in other notable works rather small content catalogs have been considered even in simulations (e.g.: [21, 41] consider 1000 objects, whereas [18] 2 orders of magnitude less contents), in this work we attempt to keep the number of objects we consider to a reasonable value. By considering object classes rather than single objects, we can make our optimization models scale to real-size content catalogs, as done in other works such as [33], even though this introduces a quantization error since the optimal object placement is performed with respect to the entire class, rather than the single objects themselves. To numerically quantify the impact of this quantization error, we solved a relaxed version of our proposed OAR-TR-CDN model. Assuming that links have an unbounded capacity we can find the optimal solution for a non-quantized catalog comprising up to 10^7 objects, and we compare such solution aggregating the contents into 100 popularity classes. On the other hand, to evaluate the impact of pushing the cached content to the CDN surrogates, we evaluate ex-post the cost to perform such migration. Table 6 provides a brief summary of the obtained results. The average error introduced for object quantization is always below 5%, whereas the effect of moving objects to the CDN nodes is negligible and always below 0.02%. For the scope of this paper, we

can easily ignore both these effects since they do not compromise the results we discussed in the previous sections.

5.5. Effect of Model Assumptions

In order to get a deeper understanding of the effect of the model assumptions we made in Sec. 2.3 to differentiate CCN with respect to CDN, in this section we study their impact on each of the considered topologies.

To perform such analysis, we consider three *"hybrid"* formulations (omitted here for the sake of brevity) based on our OAR-TS model for the CCN. Each of these hybrid formulations permits to study the effect of one of the characterizing assumptions, as detailed hereafter:

- **A1: Storage Size and Placement Assumption**. In this analysis we keep all the assumptions made for the CCN, but we restrict the number of caching nodes to 3, and we place them using the k-median model (as done for the CDN). We keep the same amount of overall storage, but we distribute it on fewer nodes.

- **A2: Delay Assumption**. We consider the CCN network and let each node change the set of cached objects once every 3 time slots, as if we were considering a relocation time of $|\mathcal{T}_r| = 3$.

- **A3: Content Push Assumption**. We consider the CCN network and relax constraint (21), which forces each node to cache the subset of objects it was caching or that it forwarded in the previous time slot.

In Table 7 we show the average impact of the model assumptions comparing the base CCN model with respect to these hybrid formulations.

First of all, we observe that by using localized caches with larger storage (as in *A1: Storage Size and Placement Assumption*) we obtain in every topology a performance gain, compared to the base CCN model. In particular, the Géant topology is the one that benefits more from this assumption. As expected, by introducing a delay (as in *A2: Delay Assumption*), the value of the objective

Table 7: Effects of the Model Assumptions.

Topology	**A1: Storage Size and Placement Assumption**	**A2: Delay Assumption**	**A3: Content Push Assumption**
Abilene	1.05% Avg. Gain	8.62% Avg. Loss	0.81% Avg. Gain
Airtel	4.00% Avg. Gain	8.76% Avg. Loss	4.78% Avg. Gain
Claranet	6.41% Avg. Gain	8.43% Avg. Loss	3.48% Avg. Gain
Géant	15.41% Avg. Gain	4.44% Avg. Loss	6.01% Avg. Gain
Netrail	3.94% Avg. Gain	6.76% Avg. Loss	0.82% Avg. Gain
Sprint	3.86% Avg. Gain	7.38% Avg. Loss	1.41% Avg. Gain

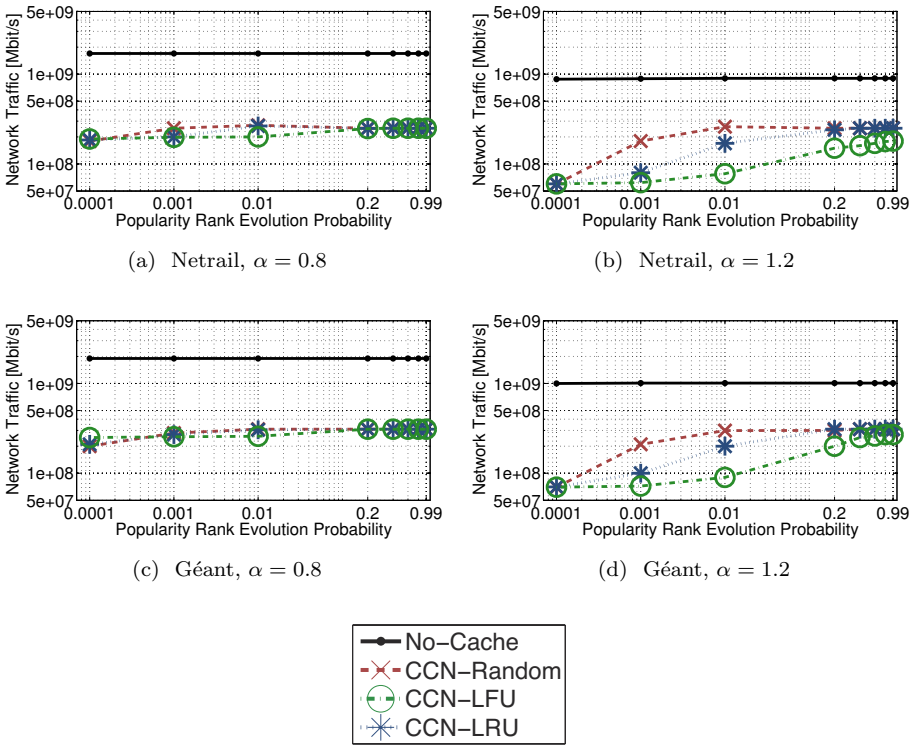
function is penalized since caches cannot quickly react to content popularity changes. Finally, in *A3: Content Push Assumption* we observe that only minor benefits are experienced when letting nodes cache any content in the catalog, rather than forcing them to store the contents they processed in the immediate past.

5.6. Simulations Numerical Results

In this section, we present the methodology and the numerical results we obtained performing simulation campaigns to further complement our analysis.

To this purpose, we extended the ndnSIM simulator [12] in order to accurately capture the behavior of a CCN architecture under the rank-shift model discussed in Sec. 3. Compared to the analysis carried out using optimization models, simulations let us study real traffic performance values, rather than their theoretical bounds. In addition to that, through the usage of simulations we can assess the performance of the Least Recently Used (LRU) cache policy that otherwise would be neglected in our analysis. To ensure consistency between the results generated for the optimization models, and those obtained through simulations, we forced the consumers to generate a constant rate traffic of 10 Mbytes/s, in line with what we did in Sec. 5.2.

Table 4 summarizes the simulation parameters used to perform our analysis. In particular, whenever possible, we used the same values adopted in the



(a) Netrail, $\alpha = 0.8$

(b) Netrail, $\alpha = 1.2$

(c) Géant, $\alpha = 0.8$

(d) Géant, $\alpha = 1.2$

(e) Legend

Figure 10: *Simulation Results.* Simulated average traffic, in a 5 minute interval, as a function of the popularity evolution parameter, under different topologies and α values. The legend 10e is common for all the plots.

optimization models, so that we can compare the results obtained with both methodologies.

We generate traffic traces of 1 hour. In order to eliminate any transient behavior, we remove the first 10 minutes from the traces, then we aggregate the traffic in intervals of 5 minutes each, and plot the average traffic value for all of them.

Figures 10a and 10b refer to the Netrail topology, whereas corresponding results for the Géant topology are depicted in Fig. 10c and 10d, respectively. First of all, the simulations confirm the same trends for the caching policies presented in the previous section. The total traffic for Géant reaches higher

values than that obtained for Netrail. This behavior is due to the fact that the Géant network has a larger network diameter than Netrail, which raises the value of the traffic metric.

Simulations further confirm that the popularity evolution parameter (ρ) does not have any impact on a network that is not implementing caching functionalities, as shown by the No-Cache curves. We also observe that by using higher Zipf α exponents, caching can further lower the total traffic exchanged in the network, since content requests will be mostly concentrated on a small subset of the available objects that will likely be replicated in the distributed caches.

An interesting finding (in line with the results presented in [34]), is that different cache replacement policies do not lead to significantly distant performance results, especially with smaller values for the α exponent, (as shown in Fig. 10a and 10b). In addition to that, we also observe that the Least Frequently Used (LFU) policy performs usually better than the others, saving up to $5 \cdot 10^7$ Mbit/s more than the Random caching policy, when $\alpha = 1.2$.

6. Related Work

This section provides a survey on Content-Centric and Content-Delivery Networks (Sec. 6.1), as well as on content popularity evolution models (Sec. 6.2).

6.1. Performance Analysis of Content-Centric and Content-Delivery Networks

Hereafter we review the most notable research works on CCN and CDN architectures, specifically focusing on the performance analysis of these networks.

Fayazbakhsh et al. study in [9] the performance of a Content-Centric Network specifically comparing the improvements achievable with an *edge-based caching* with respect to a *full-fledged* CCN. In particular, they show that most of the performance benefits can be gained by deploying caching storage only on the edge nodes, whereas by distributing the memory also on the other nodes they can achieve a limited 4% performance gain. A similar question is investigated in [21], where pervasive caching and opportunistic edge caching are compared

under a traffic model that considers both the temporal and spatial locality of requests.

Thorough simulation campaigns are presented in [34], where Rossini and Rossi study the performance of CCN focusing on forwarding strategies and caching policies. They show that limited benefits can be achieved by adopting complex caching policies: randomized caching decisions can perform as well as more complex ones, while significantly reducing the computational overhead that they introduce. Moreover, they also show that multi-path forwarding capabilities may play against the network efficiency.

Significant effort has been devoted to the performance analysis of different caching schemes. Without pretending to be exhaustive, Berger et al. study in [41] TTL cache networks, and provide analytical methods to characterize the performance of these systems. Abedini and Shakkottai consider in [18] the wireless scenario, with elastic and inelastic traffic requests. Finally, a very promising research path is the one that considers virtualized storage resources as in [42], where a cloud-based content delivery system is taken into account.

One of the scenarios where CCN can potentially boost the network performance is video distribution [35, 43, 44]. Due to the relatively small content catalog, large file sizes and the static nature of content (especially when compared to web pages), video distribution should take advantage of in-network caching capabilities of CCN, moving the content closer to the locations where most of the users are actually requesting it.

In order to satisfy video demands, nowadays other technological infrastructures known as Content-Delivery Networks (CDNs), built as overlays on top of TCP/IP, are exploited to serve this precise scope.

Adhikari et al. study in [45] the Netflix video streaming platform, by analyzing the video delivery performance in a scenario where many CDNs are used for video streaming purposes. In [46], Mansy and Ammar focus on a scenario where the CDN cooperates with P2P to serve *adaptive* video streaming requests. The authors show that such a hybrid scenario has interesting performance properties that can potentially reduce the costs paid by the content

provider. Liu et al. have measured in [38] the performance of video distribution when clients leverage the CDN infrastructure to retrieve video content. In order to further improve the users' *quality of experience*, the authors suggest to adopt a control plane that automatically selects the best bit-rate and CDN server according to the network state.

In terms of performance optimization, three classic problems have to be solved in a CDN:

1. *Server Placement*: choose the locations where to place the CDN servers.
2. *Replica Object Placement*: choose the locations and the number of object replicas by distributing them on the available servers.
3. *Surrogate Server Selection*: route object requests by selecting a replica server storing a copy of the given object.

Without pretending to be exhaustive, hereafter we mention relevant works addressing each of the above-mentioned problems. In particular, the *center placement problem* is used to solve both server placement as well as the replica object placement, by modeling the problem as a graph where a given distance metric should be minimized [11, 47]. The size of the problem, which becomes even larger when studying object placement, forces to formulate heuristic algorithms that find sub-optimal solutions, such as those presented in [48, 49].

Finally, in [50], surrogate server selection strategies used by YouTube are evaluated in order to understand which parameters may influence the CDN server selection process.

6.2. Content Popularity Models

In this section we review relevant works regarding content popularity evolution in the Internet. Due to its worldwide success, many research works have studied the popularity evolution of video contents [51, 37, 24].

In [51], Cha et al. have studied *Video-on-Demand* (VoD) systems for *User Generated Content* (UGC). The authors provide evidences that the popularity of UGC

is ephemeral, and traditional content popularity prediction techniques are ineffective. In fact, while TV-broadcasters deliver the same content to all their users at the same time, VoD services, instead, let each customer choose which video she is going to watch.

Dán and Carlsson in [37], and Cha et al. in [51] observe that an exponential cutoff term should be added to a power-law model to better represent the content popularity. In [24], Figueiredo et al. provide evidences that the popularity evolution of a given UGC video depends also on the type of content it represents. In particular, copyright-protected materials tend to get most of their views very early in the lifetime, and sudden burst of popularity are very common among top-list videos.

Despite being very interesting, all the works reviewed so far cannot be easily used to generate *synthetic workloads*; instead, two models to produce synthetic traces have been presented in [25, 26]. Borghol et al. present in [25] a model that can be used to generate such synthetic workload to represent the popularity evolution of UGC. Their model splits the lifetime of a content in three stages: *before*, *at* or *after* the age at which the content reaches its peak popularity. The main shortcoming of this model is that it requires several parameters to be used in practice:

1. the content popularity distribution for each stage of the lifetime; 2. the content movement process across the different stages and 3. the consumer view rate distributions for contents belonging to each group.

Another model, that we adopted in this paper to generate realistic workloads in the presence of bursty popularity evolution, has been presented by Ratkiewicz et al. in [26], and has been extensively discussed in Sec. 3. Despite the fact that it models the popularity evolution with only one parameter, this proposal has many strengths: it is accurate (as shown by the authors themselves [26]) and, at the same time, simple.

7. Conclusion

In this paper we compared CCN and CDN by formulating novel optimization models to analyze the performance bounds of these network architectures, considering time-varying demands to represent content popularity evolution. The proposed models are such that the relevant characteristics of CCN and CDN are explicitly taken into account.

Our numerical results suggest that the performance bounds for CDN architectures are better in terms of network traffic than observed for CCN, even when few CDN replica servers are deployed in the network. In our considered scenarios, we observed that in the Netrail topology CCN can reach a better value for the objective function compared to CDN only if this latter is at least 15 times slower to react to popularity changes. On the other hand, while considering the larger Géant topology, we observed that CDN is always to be preferred since it reaches up to 14% better performance than CCN.

Our analysis is in line with the results presented in [9]: spreading the caching storage uniformly in the network does not necessarily lead to better performance than that obtained by concentrating it on fewer nodes. In other words, in terms of the overall network traffic CCN does not necessarily perform better than CDN. Our view is that rather than being two antagonist models, CCN and CDN should complement each other; in particular CCN is a very promising architecture to reduce the management overhead that the network introduces, whereas CDN is to be preferred if the main goal is to achieve the highest performance gains.

References

[1] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, M. Varvello, From content delivery today to information centric networking, Computer Networks 57 (16) (2013) 3116 – 3127.

[2] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, F. E. Bustamante, Drafting

Behind Akamai: Inferring Network Conditions Based on CDN Redirections, IEEE/ACM Trans. on Networking (TON) 17 (6) (2009) 1752–1765.

[3] J. Kurose, Information-Centric Networking: The Evolution from Circuits to Packets to Content, Computer Networks 66 (2014) 112–120.

[4] J. H. Saltzer, D. P. Reed, D. D. Clark, End-to-end arguments in system design, ACM Trans. on Computer Systems (TOCS) 2 (4) (1984) 277–288.

[5] Cisco Press Release - Cisco Adds Carrier Routing System X (CRS-X) Core Router to Industry-Leading CRS Family, San Jose, USA, June, 2013, `http://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1208192`, Last accessed: November 2015.

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

[7] E. Nygren, R. K. Sitaraman, J. Sun, The Akamai network: a platform for high-performance internet applications, ACM SIGOPS Operating Systems Review 44 (3) (July 2010) 2–19.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, Networking Named Content, in: Proc. of the 5th Int.l Conf. on Emerging networking experiments and technologies (CoNEXT), ACM, Rome, Italy, December 2009, pp. 1–12.

[9] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, S. Shenker, Less Pain, Most of the Gain: Incrementally Deployable ICN, in: Proc. of the ACM SIGCOMM conference, Hong Kong, China, August 2013, pp. 147–158.

[10] D. Rossi, G. Rossini, On Sizing CCN Content Stores by Exploiting Topological Information, IEEE INFOCOM, NOMEN Workshop (Orlando, Florida, (USA), March 2012) 280 – 285.

[11] L. Qiu, V. N. Padmanabhan, G. M. Voelker, On the placement of web server replicas, in: Proc. of IEEE INFOCOM Conference, Anchorage, Alaska, (USA), April 2001, pp. 1587–1596.

[12] A. Afanasyev, I. Moiseenko, L. Zhang, ndnSIM: NDN simulator for NS-3, Technical Report NDN-0005, NDN (October 2012).

[13] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica, A Data-Oriented (and Beyond) Network Architecture, SIGCOMM Computer Communication Review 37 (4) (2007) 181–192.

[14] N. Fotiou, P. Nikander, D. Trossen, G. C. Polyzos, Developing Information Networking Further: From PSIRP to PURSUIT, in: Broadband Communications, Networks, and Systems, Vol. 66 of LNCS, Social Informatics and Telecommunications Engineering, Springer, 2012, pp. 1–13.

[15] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, L. Correia, The way 4WARD to the Creation of a Future Internet, in: Proc. of IEEE 19th Int.l Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, Cannes, France, September 2008, pp. 1–5.

[16] M. Adler, R. K. Sitaraman, H. Venkataramani, Algorithms for Optimizing the Bandwidth Cost of Content Delivery, Computer Networks 55 (18) (2011) 4007–4020.

[17] G. Pallis, A. Vakali, Insight and perspectives for content delivery networks, Commun. ACM 49 (1) (2006) 101–106.

[18] N. Abedini, S. Shakkottai, Content caching and scheduling in wireless networks with elastic and inelastic traffic, IEEE/ACM Transactions on Networking 22 (3) (2014) 864–874. doi:10.1109/TNET.2013.2261542.

[19] L. Wang, J. Kangasharju, Fair Collaborative In-Network Caching Game and Its Cost Analysis on General Network Topologies, Tech. rep. (Nov. 2014). arXiv:1412.0041.

[20] G. Zhang, Y. Li, T. Lin, Caching in information centric networking: A survey, Computer Networks 57 (16) (2013) 3128–3141. doi:10.1016/j.comnet.2013.07.007.

[21] A. Dabirmoghaddam, M. M. Barijough, J. Garcia-Luna-Aceves, Understanding optimal caching and opportunistic caching at "the edge" of information-centric networks, in: Proc. of the 1st Int.l Conf. on Information-centric networking - ICN '14, ACM Press, Paris, France, 2014, pp. 47–56. doi:10.1145/2660129.2660143.

[22] D. Xu, S. S. Kulkarni, C. Rosenberg, H. K. Chai, Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution, Multimedia Systems 11 (4) (2006) 383–399. doi:10.1007/s00530-006-0015-3.

[23] D. O. Coileáin, D. O'mahony, Accounting and Accountability in Content Distribution Architectures, ACM Computing Surveys 47 (4) (2015) 1–35. doi:10.1145/2723701.

[24] F. Figueiredo, F. Benevenuto, J. M. Almeida, The Tube Over Time: Characterizing Popularity Growth of Youtube Videos, in: Proc. of the 4th ACM Int.l Conf. on Web Search and Data Mining (WSDM), Hong Kong, February 2011, pp. 745–754.

[25] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, A. Mahanti, Characterizing and Modelling Popularity of User-generated Videos, Performance Evaluation 68 (11) (2011) 1037–1055.

[26] J. Ratkiewicz, F. Menczer, S. Fortunato, A. Flammini, A. Vespignani, Traffic in Social Media II: Modeling Bursty Popularity, in: Proc. of the 2nd Int.l IEEE Conf. on Social Computing (SOCIALCOM), Minneapolis, MN, USA, August 2010, pp. 393–400.

[27] G. Carofiglio, V. Gehlen, D. Perino, Experimental Evaluation of Memory Management in Content-Centric Networking, in: Proc. of the IEEE Int.l Conf. on Communications (ICC), Kyoto, Japan, June 2011, pp. 1–6.

[28] G. Pallis, Improving Content Delivery by Exploiting the Utility of CDN Servers, in: Proc. of the 5th Int.l Conf. on Data Management in Cloud, Grid and P2P Systems (Globe), LNCS, Springer, Vienna, Austria, September 5-6, 2012, pp. 88–99.

[29] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: Evidence and implications, in: Proc. of IEEE INFOCOM, Vol. 1, New York, USA, March '99, pp. 126–134.

[30] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, K. K. Ramakrishnan, Optimal content placement for a large-scale VoD system, in: Co-NEXT, ACM, Philadelphia, Pennsylvania, USA, 2010, pp. 1–12. doi:10.1145/1921168.1921174.

[31] Géant Network Website, http://geant3.archive.geant.net/Network/NetworkTopology/pages/home.aspx, Last accessed: November 2015.

[32] P. Van Hentenryck, The OPL optimization programming language, Mit Press, 1999.

[33] L. Gkatzikis, V. Sourlas, C. Fischione, I. Koutsopoulos, G. Dán, Clustered Content Replication for Hierarchical Content Delivery Networks, in: IEEE International Conference on Communications (ICC), IEEE, London, UK, 2015, p. 6.

[34] G. Rossini, D. Rossi, Evaluating CCN multi-path interest forwarding strategies, Computer Communications 36 (7) (2013) 771 – 778.

[35] Z. Li, G. Simon, Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching, in: Proc. of the IEEE International Conf. on Communications (ICC), Kyoto, Japan, June 2011. doi:10.1109/icc.2011.5963380.

[36] Repository and Complementary Results, Available at: https://www.lri.fr/~mangili/ccn-cdn.html, Last accessed: November 2015.

[37] G. Dán, N. Carlsson, Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity, in: Proc. of the Int.l Workshop on Peer-to-Peer Systems (IPTPS), San Jose, CA, USA, April 2010.

[38] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, H. Zhang, A case for a coordinated internet video control plane, in: Proc. of ACM SIGCOMM, Helsinki, Finland, August 2012, pp. 359–370.

[39] C. Fricker, P. Robert, J. Roberts, N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, IEEE INFOCOM, NOMEN Workshop (Orlando, Florida (USA), March, 2012) 310–315.

[40] K. Poularakis, G. Iosifidis, L. Tassiulas, Approximation caching and routing algorithms for massive mobile data delivery, in: IEEE Global Communications Conference (GLOBECOM), IEEE, Atlanta, GA, USA, 2013, pp. 3534–3539. doi:10.1109/GLOCOM.2013.6831621.

[41] D. S. Berger, P. Gland, S. Singla, F. Ciucu, Exact analysis of TTL cache networks, Performance Evaluation 79 (2014) 2–23. doi:10.1016/j.peva.2014.07.001.

[42] N. Carlsson, D. Eager, A. Gopinathan, Z. Li, Caching and optimized request routing in cloud-based content delivery systems, Performance Evaluation 79 (2014) 38–55. doi:10.1016/j.peva.2014.07.003.

[43] J. Choi, J. Han, E. Cho, T. Kwon, Y. Choi, A Survey on Content-Oriented Networking for Efficient Content Delivery, IEEE Communications Magazine 49 (3) (2011) 121–127. doi:10.1109/MCOM.2011.5723809.

[44] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, H. Hellwagner, An experimental analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks, in: Proc. of IEEE Conf. on Multimedia and Expo (ICME), San Jose, California, (USA), July 2013, pp. 1–6.

[45] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z.-L. Zhang, Unreeling netflix: Understanding and improving multi-CDN movie

delivery, in: Proc. of IEEE INFOCOM, Orlando, Florida, (USA), March 2012, pp. 1620–1628.

[46] A. Mansy, M. H. Ammar, Analysis of adaptive streaming for hybrid CDN/P2P live video systems, in: Proc. of the 19th IEEE Int.l Conf. on Network Protocols ICNP, 2011, pp. 276–285.

[47] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, On the placement of Internet instrumentation, in: Proc. of IEEE INFOCOM, Vol. 1, Tel-Aviv, Israel, March 2000, pp. 295–304.

[48] P. Krishnan, D. Raz, Y. Shavitt, The cache location problem, IEEE/ACM Trans. on Networking (TON) 8 (5) (2000) 568–582. doi:10.1109/90.879344.

[49] S. Jamin, C. Jin, A. Kurc, D. Raz, Y. Shavitt, Constrained mirror placement on the Internet, in: Proc. of IEEE INFOCOM, Vol. 1, Anchorage, Alaska, (USA), April 2001, pp. 31–40.

[50] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, S. Rao, Dissecting Video Server Selection Strategies in the YouTube CDN, in: Proc. of IEEE Conf. on Distributed Computing Systems (ICDCS), Minneapolis, Minnesota, (USA), June 2011, pp. 248–257.

[51] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems, IEEE/ACM Trans. on Networking 17 (5) (2009) 1357–1370.