Caching performance of content centric networks under multi-path routing (and more)

Dario Rossi, Giuseppe Rossini
Telecom ParisTech, Paris, France – `first.last@enst.fr`

Abstract—**This work addresses the performance evaluation of Content Centric Networks (CCN). Focusing on a realistic YouTube-like catalog, we conduct a very thorough simulation study of the main system performance, consider several ingredients such as network topology, multi-path routing, content popularity, caching decisions and replacement policies.**

Summarizing our main results, we gather that (i) the impact of the topology is limited, (ii) multi-path routing may play against CCN efficiency, (iii) simple randomized policies perform almost as well as more complex ones, (iv) catalog and popularity settings plays by far the most crucial role above all. Hopefully, our thorough assessment of scenario parameters can assist and promote the cross-comparison in the research community – for which we also provide our CCN simulator as open source software.

I. Introduction

Though some might argue that “even on the Internet, content is not king” [1] still applies today, a new communication paradigm, namely Information centric networking (ICN), may have a disruptive impact on the Internet ecosystem. The adoption of ICN may indeed not only reshape the underlying Internet technology, but also threaten the current business models, with the content and optimization business moving down from “over the top” approaches such as Content Distribution Networks (CDN) to lower layer services, available directly to the owner of the infrastructure.

Recognizing that end users are often more interested in obtaining *content*, rather than merely being provided with *connectivity* among two addressable entities, a number of architectures (overviewed in [2]) have started proposing caching of objects (or object chunks) as network primitives. While these proposal differ in a number of aspects (e.g., the way content is named, content resolution is addressed, etc.) the crucial importance of *in-network caching of popular content* is common to all. Among these proposals, Content Centric Networking (CCN) [3] is one the most promising: in CCN, content is sent in reply to *interest packets* addressing *named data chunks*, that gets cached along the way back to the request originator. In CCN routers, caches de facto replace traditional buffers, so to avoid redundant network traffic [4]: interests packets are matched with cached data, which has to be done per packet, for all requests, and at line speed.

Yet, the potential impact of CCN has not been thoroughly assessed so far: though caching has already been extensively studied, a number of architectural details make caching in CCN a relatively new, and largely unexplored, research topic. The lack of CCN performance evaluation is partly due to the scale of the problem (large CCN cache and Internet catalog sizes), to its strict requirements (line speed operation) and complex scenarios (network of caches, user behavior, etc.). For one, as in CCN all nodes may cache the content, and since multiple paths can be used, it follows that the network of caches is no longer arranged as a tree, but as an arbitrary graph. Moreover, though CCN is expected to offload a significant amount of traffic from ISP networks, we stress that ISPs are interested in offloading their peering links with other ASs, rather than their internal links. Otherwise stated, ASs will be interested in fetching new CCN data chunks only once from external ASs, to reduce the transit traffic: this translates into the need for maximizing the content diversity of caches within an AS, so that most of the content can be found within the AS boundaries. Notice that this may partly tradeoff with the minimization of user delay and network bandwidth usage – as simply caching many replicas of popular content, the nearest possible to the user, may reduce cache diversity. Finally, contrary to early literature on Web caching, there is no consensus on the evaluation scenario for CCN in the scientific community so far – which is exacerbated by the changing nature of Internet applications and users’ behavior [5].

The contribution of this paper are as follows. First, we develop an efficient and scalable chunk-level CCN simulator, that we make available to the scientific community as open source software [6]. The simulator allows us to assess CCN performance in scenarios with large orders of magnitude for CCN cache sizes (up to 10^6 chunks), catalog sizes (up to 10^8 files), and file sizes (up to 10^3 chunks). Second, we conduct a thorough simulation campaign, considering several popularity settings, 6 topologies, 2 routing strategies, 4 cache decision and 4 cache replacement policies. Results of this campaign allow us to conclude that (i) the impact of the topology is limited, (ii) multi-path routing may play against CCN efficiency, (iii) simple randomized policies perform almost as well as more complex ones, (iv) catalog and popularity settings plays by the way the most crucial role. Third, we take an extreme care in the scenario definition, investigating the effect of wide parameter ranges that, along with the simulator software, can hopefully assist and promote cross-comparison in the research community.

The reminder of this paper is organized as follows. Sec. II provides a thorough survey, taxonomy and comparison among the most relevant related work. Sec. III then describes the system parameters and scenarios, whose simulation results are presented in Sec. IV, after which Sec. V concludes the paper.

II. Related work

As this work focuses on *performance* aspects of CCN, for reason of space, we invite the reader to [2] and [3] for *architectural* aspects of ICN and CCN respectively (of the latter, we however provide a brief overview in Sec. III). A summary of the most relevant related work is presented in Tab. I, considering different taxonomic criteria. As previously introduced, despite a large caching literature (see e.g., [7] for a thorough survey), novel aspects of ICN architectures (e.g., splitting content in chunks, complex topologies, line speed requirement, etc.) refuel the research interest.

A first limit of current caching work is that, with few exceptions [2], [8]–[12], *entire objects* are generally cached. In CCN, content is instead partitioned in sub-objects (or *chunks*), so that different chunks of the same object may end up being cached on different CCN routers. This design decision partly follows from the efficiency of chunk-based diffusion shown by BitTorrent in P2P networks, that lately was brought into CDN [13] as well.

Concerning caching policies, as pointed out in [14], much attention has been devoted to *cache replacement* issues (over 40 policies are overviewed in [7]), while *cache decision* policies have being studied sporadically (to the best of our knowledge, only by [14]–[17]). Still, due to the fact that caching operations must happen at line speed in CCN routers, most of the existent decision and replacement policies are not of practical relevance, as [10] points out.

As far as *replacement policies* are considered, Least Recently Used (LRU) has been used in the context of CCN [8], [11], [12] and of the more general ICN context [2], [9], [18]. Only few work considers other policies, such as Most Recently Used (MRU) and Most Frequently Used (MFU) in ICN [9]. Interestingly, [9] shows LRU performance to be indistinguishable from MRU/MFU, which are thus not sufficient to counter the “dictatorship of popularity”. Moreover, even a simple LRU policy –often used in turn as a good enough approximation of Least Frequently Used (LFU)– may be too complex to implement, so that Uniform random replacement (UNIF) may be preferable to keep CCN simple and scalable. Possibly, $N > 2$ elements are sampled for eviction, and after the replacement decision is taken M out of the $N-1$ remaining elements are kept for the next eviction, as this may provide significant performance improvements [19].

As far as *decision policies* are considered, generally the assumption is made that any new content gets always cached. Few exceptions to this rule come from the Web caching [14]–[17] or CCN [10] contexts: however, the approach in [15] doesn't apply to CCN due to implementation complexity, while DEMOTE [16] is known to poorly perform on network of caches. With this respect, only the Leave Copy Down (LCD) policy [14], [17] is simple enough to be worth implementing in CCN. Again, an even simpler policy is considered by [10], where caching decisions are taken uniformly at random with a fixed probability FIX(P).

Finally, we point out that *explicit cache coordination* policies (e.g., see [20] for Web, and [21] for ad hoc domain) would likely violate CCN line of speed constraint. For instance, [21] estimates the local density of requested chunks, in order to decide if they are worth caching (and for how long). Density estimation is performed during an inference phase for each chunk, that would introduce a per-chunk slowdown that is simply not affordable in the CCN context.

TABLE I
COMPARISON OF RELATED WORK ON CACHING FOR WEB, ICN AND CCN

Granularity:	Object-level caching chosen by most work, except chunk-level [2], [8]–[12]
Replacement:	LRU [2], [8], [9], [18], MRU, MFU [9] UNIF [10], UNIF(M,N) [19]
Decision:	Content always implicitly cached, except FIX(P) [10], [14], [17], LCD [14], [17]
Topology:	Tandem and cascade [10]–[12], [14], [17] Trees [8], [11], [12], [14], [17], synthetic GT-ITM stub with 310 [2] or 1225 [9] nodes.

We now stress what are, in our opinion, the limit of existing work and why they are not fully faithful of CCN performance in an Internet environment, considering (i) analytical and (ii) simulation based studies. First, most of previously developed models are based on different assumptions that are not fit to CCN due to either (i) chunks partitioning or (ii) topological assumption. As CCN requests for consecutive chunks of the same objects are now correlated, the independent reference model (IR, where all requests are i.i.d.), popular in caching studies, no longer holds: correlated arrivals are only studied in recent work [11], [12], though the analysis is limited to simple cascade or tree topologies. Conversely, though the approximate cache model in [22] applies to general network topologies, it considers an object granularity (and further assumes that on a cache miss, the object is instantaneously downloaded before the next request for the same object hits the cache). Hence, to the best of our knowledge, an analytical work overcoming both limits has yet to appear.

Yet, even simulation-based studies of information-oriented networking are often simplistic in their (i) topological assumptions or due to the (ii) scale of the considered system. Indeed, with the exception of [2], [9] (employing synthetic topologies generated with GT-ITM), most simulative work still considers simple topologies (e.g., cascade or trees [8], [10]–[12]).

Even more important, the scale of the considered system is often underestimated. Notice indeed that, in Web caching it was reasonable to assume pretty large amount of caches (e.g., disks), which implied that an accurate estimate of the ratio between the cache size and the catalog size was not an issue (e.g., add disks). In CCN instead, cache size is technologically limited by memory access speed [10], due to the fact that interest and data packets need to be serviced at the Network Interface Card (NIC) speed: as it is not possible to arbitrarily increase size of caches on board of CCN routers, it becomes imperative to more accurately estimate the size of the catalog that CCN is expected to service, in order to assess whether CCN is able to really achieve the promised breakthrough.

TABLE II
SYSTEM PARAMETERS INVESTIGATED IN THIS WORK

Parameter	Meaning	Values
c	Chunk size	10 KBytes
F	File size	up to 10^4 chunks (10 MB) (geom. distributed)
$\|F\|$	File number	up to 10^8 files
$\|F\|F$	Catalog size	up to 10^15 bytes (1 PB)
C	Cache size	up to 10^6 chunks (10 GB)
$\frac{C}{\|F\|F}$	Cache/catalog ratio	$[10^{-5}, 10^{-1}]$
α	Zipf exponent	[0.5,2.5]
q	MZipf plateau	{0,5,50}
λ	Arrival rate	[1,10]Hz
$W = 1$	Control window width	1 chunk
R	Number of paths	{1,2}
C_R	Cache replacement policy	FIFO,LRU,UNIF,BIAS
C_D	Cache decision policy	ALWAYS,FIX(P),LCD
Net	Network topology	see Tab. IV

TABLE III
CHUNK, FILE, CACHE, CATALOG AND POPULARITY IN RELATED WORK

Parameter	Values
c	1KB [8], 10KB [11], [12], 16MB [9]
C	6.4MB [8], 50MB [11], 2GB [12], 10GB [10]
F	1MB [11], 7MB [12], 690MB [9]
$\|F\|$	250-500 [22] , 1K [2], [18], 20K [11], [12], 100K [17] files
$\frac{C}{\|F\|F}$	0.25% [11], 0.5%-20% [9], 1-10% [17], 1.45% [12], 4% [18], 4%-10% [22]
α	0.6 [9], 0.9 [17], 1 [2], [22], 2 [11], up to 2.5 [12]

III. SYSTEM AND SCENARIO DESCRIPTION

While for reason of space, we invite the reader to [3] for a description of the CCN architecture, we need to briefly introduce CCN terminology. CCN is based on data structures such as *Content Store (CS)*, *Pending Interest Table (PIT)* and *Forwarding Information Base (FIB)* table. These structures are consulted at each *interest packet* that users express for (univocally addressable) *named data*. More precisely, a CS lookup is performed for each interest packet in the data plane: in case of a cache miss, the interface of the incoming interest is appended to the PIT, and the interest packet gets routed according to FIB information. This process iterates so that, when a cache is hit, a *data chunk* is sent back in the network, and travels through the information stored in the PIT: whenever an interest is satisfied, the corresponding entry in the PIT is then removed.

Besides cache and catalog size (Sec. III-A) or content popularity (Sec. III-B), already at first sight, several details may affect CCN performance: e.g., on different *network topologies* (Sec. III-C), interest may be forwarded along either a single shortest path, or through multiple-paths, depending on the *routing algorithm* (Sec. III-D). Then, once data travels back in the CCN data plane following the PIT table, CCN routers along the path participate in a caching process, that can be modeled as being composed by two distinct policies. First, a *decision policy* (Sec. III-E) is taken whether or not to cache the current data. Then, in case the router decides to store the object, it may need to evict a chunk according to a *replacement policy* (Sec. III-F) in case the cache is full.

In the reminder of this section, we describe the parameter space and support we investigate in this work, that we summarize in Tab. II for the sake of readability.

A. Cache and catalog size, content popularity

As we previously argued, the scale of the considered system is often underestimated, or at least the operational point at which the system is evaluated is not always realistic. To convince of the importance of the system parameters related to the catalog, we have summarized in Tab. III, some of the most relevant system parameters used in related work, namely the size of (i) content chunks, (ii) objects, (iii) router caches and (iv) the whole catalog (in number of objects).

From the table one can evince that CCN chunks are expected to be packet-size (1KB [8]-10KB [11], [12]), hence much smaller than the typical chunk sizes of CDN (60KB [13]), P2P (BitTorrent 250KB-4096KB [23]) and other ICN architectures (16MB [9]). In our work, we select 10KB chunks as in [11], [12]: despite a CCN chunk size is not specified in [3], we believe that in reason of the CCN overhead –due to (i) interest packets and (ii) large headers for content naming and security issues– chunk sizes smaller than 10KB would likely be an overkill.

As far as CCN router cache size is concerned, [10] starts from the observation [4] that about half of the caching benefits at packet level happen in the first 10 sec.: considering a capacity of 1Gbps, [10] sizes to about 12 GB the amount of memory that can be addressed at line speed (adopting a constructive approach and employing a two-levels address scheme). Yet, the cache size typically considered in simulation is much smaller (from 6.4MB [8] to 50MB [11] and 2GB [12]), although no proper justification for these selections is given. Hence, we select cache sizes of 10 GB (10^{10} Bytes or 10^6 chunks) as a realistic case study of CCN performance.

Finally, the number of files in the considered catalogs can be as low as 250 objects [22], topping to 20K [11], [12] objects for the largest CCN catalogs. Taking into account also the object size (in chunks) considered in those work, largest catalogs varies between 2Mchunks [11] and 13.8Mchunks [12], i.e., 20GB and 138GB respectively. Yet, we believe these sizes to be extremely small compared to Internet catalogs – which is easily confirmed by summing up the storage size of our portable devices.

To gather a more realistic estimate of nowadays Internet catalog size, we consider two of the most popular Internet content diffusion systems: namely, the YouTube portal for video content, and the BitTorrent P2P file-sharing application. To estimate the catalog size, we consider the approximate number of videos (about 10^8 [24]) and torrents (roughly, $5 \cdot 10^6$ [25]) and their respective average object size (10MB [26]) and (760MB [27]): overall, we can estimate both catalogs to be on the order of a few PB (i.e., 10^{15} Bytes). Notice that, since the above values have been gathered at different times, from different vantage points, on different sets, we expect to get only a rough estimate of the catalog size. Yet we see that previous work underestimated the catalog size of about 4 orders of magnitude. In the following, we consider a YouTube-like catalog, consisting of 10^8 objects (100M), of geometrically distributed[1] size with average 10^3 chunks (10MB).

Moreover, we expect system performance to be determined by the ratio of the cache size C over the catalog size $|F|F$. As reported in Tab. III, the ratio $\frac{C}{|F|F}$ considered in the ICN literature varies between a minimum of 0.25% [11]-0.5% [9] to a maximum of 10% [22]-20% [9]. Yet, considering realistic CCN cache size and Internet catalog, this ratio seems more likely to be on the order of 10^{-5}, hence much smaller than the one considered in previous studies. As a consequence, ICN benefits may therefore be overestimated by current literature.

B. Content Popularity

Another very important aspect that concur in determining the system performance is the content popularity model. Rather typically [11], [12], [17], [18], CCN studies consider (variants of) Zipf popularity, which is simply tuned by a single parameter α, (i.e., the exponent characterizing the distribution). Yet, no consensus has been reached on the exact model, as for instance [9] resorts to a Mandelbrot-Zipf (MZipf for short) distribution, while [18] also includes uniform popularity, and [11], [12] consider several classes distributed with Zipf popularity (with objects within each class uniformly popular).

Moreover, no consensus has been reached on the actual settings either, as the considered value of α varies in a rather wide range 0.6 [9]-2.5 [12] as shown in Tab. III. For instance, the minimum of 0.6 [9] is taken from a MZipf fitting of Gnutella catalog [28] – where, in this case, the distribution is also determined by the plateau parameter ($q \in [3, 121]$ depending on the Autonomous System under observation, and typically $q \leq 50$). Values of $\alpha \in [0.6, 1.2]$ are adopted in [2], [12], [17], [22], where the extremes are typical of lightly loaded and busy Web servers respectively [29], while large values of α such as 2 [11] and 2.5 [12] derive from the very same analysis of YouTube [24] we used early to determine a realistic Internet catalog size.

Hence, the range of α values are, at least apparently, well motivated. Yet, if we consider more closely the YouTube popularity, for the sake of the example, we see that the $\alpha \geq 2$ value reported in [24] only fits part of the tail, while the body of the distribution appears more likely to follow a Mandelbrot-Zipf law[2].

Though this might seem, at a first sight, only a minor issue, in reality it is not. The α parameter of the Zipf distribution significantly shapes the request arrival pattern, practically limiting the request to a (possibly very narrow) subset of the whole catalog. Let us evaluate the rank $|F_{99}|$ of the 99-th percentile for different values of the MZipf catalog size $|F| \in \{10^4, 10^8\}$, and plateau[3] $q \in \{0, 5, 50\}$ parameters, which we depict in Fig. 1 as a function of the MZipf exponent $\alpha \in [0.5, 2.5]$. Intuitively, the 99-th percentile rank represents the number of objects in the catalog that make up the 99% bulk of users' requests.

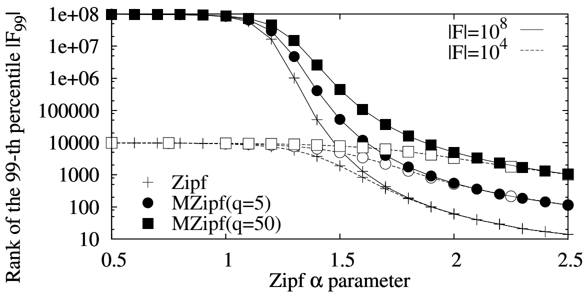


Fig. 1. Rank of the 99-th percentile of popular requests, modeled according to a MZipf distribution with catalog size $|F|$, exponent α and plateau q.

From the picture, it is easy to grasp that for $\alpha \leq 1$, almost the whole catalog needs to be cached to satisfy 99% of the requests *irrespectively of the plateau q parameter*: in this operational region, we expect caching to be hardly effective, due to the unfavorable 10^{-5} cache/catalog ratio early estimated.

Considering the Zipf case, for high values of $\alpha \geq 2$ the 99-th rank converges to the same value *irrespectively of the catalog size*. Moreover, even in the case of very large catalogs consisting of 10^8 objects, 99% of the requests are directed to slightly more than a dozen of objects. Hence, we get that if $\alpha = 2.5$ would hold for the YouTube catalog, a 150MB cache would suffice to store the bulk of the popular objects. Considering the MZipf case, notice that though the curves still converge for high α irrespectively of the catalog size, a higher number of objects needs to be cached to satisfy 99% of the requests (roughly, $|F_{99}|$ grows by one order of magnitude for each step from $q = 0$ to $q = 5$ and $q = 50$).

Hence, we expect CCN performance to be drastically determined by the popularity exponent α (and by the plateau q at a second order). Yet, as there is no consensus so far on the settings of the above parameters, rather than sticking to a specific parameter choice, in the following we explore a wide range of α and q values (as reported in Tab. II) so to assess the boundaries of CCN usefulness. As reference, we will also consider $\alpha = 1.5$ as the centerfold of the $[0.5, 2.5]$ range, and as it corresponds to the phase transition shown in Fig. 1.

C. Network topology

To promote cross comparison we resort to real network topologies that are publicly available through Rocketfuel [30], so that alternative approaches can be compared on the very same set of topologies. As a reference, we also consider a standard binary tree topology with 15 nodes and 8 leafs

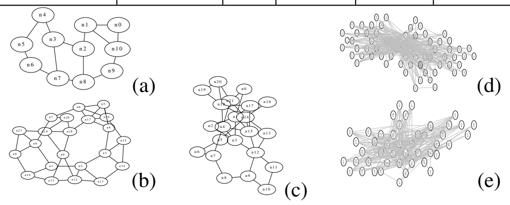
[1]Notice that YouTube filesize in [26] is well fitted by a geometric distribution (with average 10^3 chunks, of length 10KB, per file), as this reflects natural differences in the video durations. Conversely, geometric fitting of BitTorrent content would not be valid, due to peaks for CD and DVD-sized objects [27].

[2]We have not access to the popularity data shown in Fig.3 of [24] to perform a fitting, however the dissimilarity from a Zipf shape (and the similarity with Mandelbrot-Zipf) is evident.

[3]Notice that MZipf with $q = 0$ degenerates into a Zipf distribution.

TABLE IV
TOPOLOGICAL PROPERTIES OF THE CONSIDERED NETWORK TOPOLOGIES.

	Segment	N	δ	σ_δ/δ	$\Delta[ms]$	D
(a) Abilene	Core	11	2.54	0.19	11.3	8
(b) Tiger2	Metro	22	3.60	0.17	0.11	5
(c) Geant	Aggr	22	3.40	0.41	2.59	4
(d) DTelekom	Core	68	10.38	1.28	17.21	3
(e) Level3	Core	46	11.65	0.86	8.88	4
Tree	Ref.	15	2.54	0.21	1.00	3



(while we defer the use of GT-ITM topology for future work). The main characteristics of the considered topologies, corresponding to different network segments, having different sizes and properties (e.g., mean δ and standard deviation σ_delta of the degree, link delay Δ, diameter D, etc.) are reported in Tab. IV.

For each topology, we only consider the AS backbone by eliminating all single homed nodes (this is with no consequence, as the last CCN edge router is expected to serve multiple clients). We consider that the system operates at a load well below congestion: hence, as propagation delays of Tab. IV dominate transmission delays (in the range of expected capacities of CCN routers), we consider infinite backbone link capacity.

D. Routing Process

Another aspect that might affect system performance, and that makes CCN peculiar with respect to other caching studies, is that nodes are arranges with arbitrary topologies. Hence, unlike in hierarchies of Web objects, named content in CCN can in principle take *any* path in the network.

We investigate the impact of routing without however being bound to a specific routing algorithm. We therefore assume that an external routing process (and name resolution scheme) provide *multi-path alternatives* from the content requester to the content originator. Clearly, the paths of interest packets may end up earlier, in case the corresponding data is hit from the cache of any node along these paths.

As *primary* path, we consider the classical Dijkstra shortest path between the requester and the originator. As *secondary* path, we instead use [31] to find the shortest path that is the most diverse from the primary[4]: notice that by reducing the share of faith between paths, ISPs are robust face to both failures and traffic spikes.

Although multiple policies could be envisioned for the use of alternate paths in the interest/data plane (e.g., per-object vs per-chunk, alternating between paths, using the secondary path with a fixed probability, etc.) in this paper we limitedly consider two simplest alternatives – where interests follow either a single path ($R = 1$) or where both paths are used in parallel ($R = 2$).

[4]We compute the secondary path by running Dijkstra on a modified graph in which the cost of links along the primary path are increased by the network diameter, and other link costs are unitary: in this way, overlaps are tolerated when strictly necessary, i.e., when the path would otherwise be disconnected [31].

E. Cache Decision Policy

As previously introduced, cache decision policies are much less studied w.r.t replacement policies. At first sight indeed, it may appear that *all* arriving content needs to be cached, implicitly assuming recency of the arrival as a good estimator of the chunk value (i.e., the fact that the content will soon be used again, which kind of follows a LRU reasoning). However, at the network scale, systematically caching all chunks may translate in a low cache diversity, hence reducing CCN efficiency. Rather, as already noticed in [10] and [17] for cascades and tree topologies respectively, increasing the cache diversity may be beneficial to the overall system performance. In light of the above observation, we implement the following decision policies, that attempt at an uncoordinated distributed diversification of CCN router caches:

- ALWAYS: new content is always cached.
- FIX(P): caching decisions are taken uniformly at random as in [10], with a fixed probability $P \in \{0.75, 0.9\}$.
- LCD: on each miss, the Leave a Copy Down policy [17] moves the content a single hop down, but downstream router further away do not cache the request.

Intuitively, FIX(P) achieves the simplest possible level of uncoordinated and distributed cache diversification: yet, simple random decision still allow a given amount of overlap between caches. Instead, LCD moves the content from the source toward the client only on subsequent requests: hence, differently from ALWAYS, under LCD the popular content does not instantaneously replicate on all CCN routers along a path.

F. Cache Replacement Policy

Though cache replacement policies have been long studied, two peculiarities in CCN may however question previous work: first, requests are correlated (for subsequent chunks of the same object) and second, replacement must happen at line-speed (which significantly limits the variety of policies that can be implemented in practice). In this work, we consider:

- LRU: the least recently used chuck is replaced.
- FIFO: the oldest inserted chunck is replaced.
- UNIF: a chunk selected uniformly at random is replaced.
- BIAS: two chunks are selected at random, and the most popular is replaced.

Notice that we consider LRU as an upper bound for the complexity of the policies that can actually be implemented at line speed. Hence, while it is known that frequency based policies outperform LRU, they are out of CCN scope due to implementation complexity issues.

We then consider UNIF and FIFO as examples of a simple replacement policies. We point out that FIFO was the one and only cache replacement policy initially implemented in

PARC CCNx [32] prototype, while UNIF is known to work well in practice (for instance [33] show that in P2P networks, in presence of highly skewed content, LRU and UNIF tend to behave similarly).

We finally design a new, counter intuitive, cache replacement policy that selects chunks to be replaced *proportionally to their popularity*. Indeed, though it has been proven [34] that, for paging equal size objects under the IR model, the optimal caching strategy demands the static replication of the most popular documents up the capacity of the cache, however IR no longer holds in a CCN network with correlated arrivals and multiple paths. Hence, we advocate the need to re-explore the replacement policies design space.

The idea is that, as popular content will be often requested, individual caching decisions will be taken more often that for unpopular content. In turn, this will force multiple copies of popular chunk to be disseminated in many caches, to the detriment of cache diversity, which we counter by biasing the replacement toward popular chunks. In practice, this policy can be implemented with frequency counters H_i, incremented by one unit at each cache hit: whenever a new chunk arrives, we select two chunks x, y at random and replace the one with the largest hit value (i.e., $\text{argmax}_{i\in\{x,y\}} H_i$).

This policy leverages on the "power of two" principle [35], which was already used in the context of Web caching [19]. Notice that [19] proposes to keep part of the samples for the next eviction as this translate into a significant performance gain. This can be implemented by keeping the chunk with the lowest hit (i.e., the one that has not been evicted), and sampling only an additional element on the next eviction: yet, due to our selection metric H_i, an undesirable effect happens. Consider the case where we sample the least popular chunk: as this chunk is the least popular, all other counters will be incremented more frequently than this one, that will thus stay in the cache forever. For the time being, we leave refinements of this simple policy (e.g., how to keep old samples without side effects) for future work.

IV. SIMULATION RESULTS

We implement a simplified chunk-level CCN simulation model that faithfully represents the CS, PIT and FIB data structures, implements CCN operations such as interest aggregation and allows to univocally address content chunks. However, aiming at building an highly scalable and efficient simulator, we do not take into account the full details of CCN naming and security aspects [3]. The simulator, written in C++ under the Omnetpp [36] framework, allows us to assess CCN performance in scenarios with large orders of magnitude for CCN cache sizes (up to 10^6 chunks), catalog sizes (up to 10^8 files), and file sizes (up to 10^3 chunks). To promote cross-comparison in the scientific community, we make the simulator available as open-source software at [6].

In this section, we report the most interesting results obtained from a thorough simulation campaign, consisting of more than 10,000 simulations exploring over 1,000 individual system parameter settings. To gather performance metrics of interest, we operate as follows. At time $t = 0$ we run the centralized path discovery algorithm, that yields a set of multiple paths between any two node pairs. Starting from empty caches, we simulate the system until caches fills up, at which point we start the collection of all statistics, that continues until the cache hit metric converges to a stationary value. Unless otherwise stated, each simulation point reported in the following represents the average value gathered over 10 simulation runs (with error bars reporting the standard deviation).

A. Performance Metrics

Caching performance are usually expressed in terms of the hit/miss probability. In CCN networks, additional metrics are needed in order to capture the network-wide perspective, and to further qualify the cache diversity vs link usage tradeoff.

As user centric-metric, beyond the usual *cache hit* probability, we consider the *path stretch* $d/|P|$ as the number of CCN backbone hops d that the data chunk has actually traveled in the network, normalized over the path length $|P|$ until the content originator (i.e., without caching). Notice that (i) we measure the stretch only for cache hits, (ii) $d = 0$ when users find the content at the edge CCN router, and (iii) in reason of the normalization, path stretch is directly comparable across heterogeneous topologies.

Since we consider to operate in a non-congested regime, link utilization is not a relevant metric. Rather, we point out that path stretch already indirectly reveals the load level since, in a sense, it expresses the reduction of the number of links that chunks have traveled along. Still, it may be worth identifying whether possible resource hotspots may arise, such as CCN nodes that are significantly more loaded than others. To do so, we measure *load fairness*, for both data and interest packets: denoting by D_j (I_j) the amount of data chunks (interest packets) that transits from node j, we express fairness with the usual Jain formulation as $\mathcal{J}_D = \frac{(\sum_i D_i)^2}{N\sum_i D_i^2}$, with N the number of nodes in the network (low values of $\mathcal{J}_D$ indicate that hotspots arise).

Finally, as CCN-specific metric, we define the *cache diversity* as $\text{card}(\cup_i C_i)/\sum_i \text{card}(C_i)$, where $\text{card}(C_i)$ denotes the size (in chunks) of the i-th cache, and actually, $\sum_i \text{card}(C_i) = NC$ as we consider homogeneous caches. Cache diversity expresses the ratio of the unique content stored across all caches (i.e., without repetition) over the whole network cache size (i.e., considering the aggregate of N caches). Cache diversity can vary in the interval $[1/N, 1]$, with the lower bound achieved in the worst case, corresponding to exactly the same elements replicated in all caches. Notice that we expect cache diversity to be affected by both decision and replacement policies, and by popularity settings as well.

B. Content popularity, catalog and file size

We first consider the largest catalog size in CCN literature, $|F| = 10^4$ objects, and assess the cache hit probability on a binary tree (with 15 nodes and 8 leafs) for varying settings of MZipf popularity α, q. The root of the tree is connected to

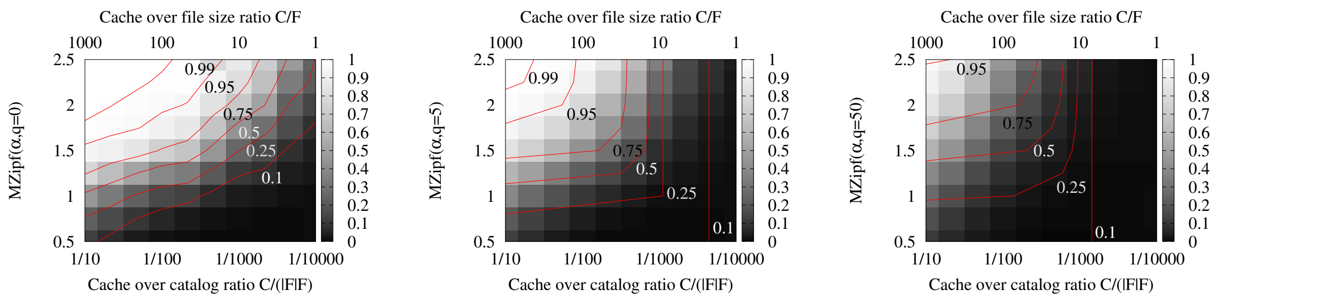


Fig. 2. Contour plot of cache hit as a function of the catalog/cache ratio (x-axis) and of the MZipf exponent α (y-axis) for different MZipf plateau q settings.

the unique repository, while the 8 leafs perform object-level requests with exponentially distributed arrival times at a 1 Hz rate. We consider standard replacement (LRU) and decision (ALWAYS) policies, and limitedly consider single path routing (due to the tree topology). File size is geometrically distributed with average $F = 10^2$ chunks, and we vary the ratio of the cache over catalog size, so that a single cache can hold between 1/10 and 1/10,000 of the whole content. Notice that performance are also affected by the average number of files that can be stored in a single cache C/F, that is reported on the top x-axis and varies from 1000 to 1.

We depict the contour plot of the cache hit probability for the above settings in Fig. 2. Notice that the explored parameter range covers almost the full support of the cache hit observable. As expected, large values of α possibly let the number of files that can be stored in the cache exceeds the 99% request percentile, which leads to an overly simple caching problem (that the plateau q may temper).

We now move to consider a large YouTube-like catalog consisting of $|F| = 10^8$ files with geometrically distributes size equal to $F = 10^3$ on average (1PB). Given that we consider $C = 10^6$ chunks large caches (10GB), the ratio $\frac{C}{|F|F} = 10^{-5}$ is thus one order of magnitude smaller than the rightmost scale of Fig. 2. At the same time, since $C/F = 10^3$ as in the leftmost scale of Fig. 2, we may expect to achieve non negligible cache hit values depending on the MZipf settings. We select $\alpha = 1.5$ as centerfold of our explored parameter range, where 99% of the requests are directed to the $|F_{99}| = 5733$ most popular files of the whole $|F| = 10^8$ catalog, about 17% of which could be stored in a single cache. Now, in a 15-nodes binary tree, all caches in the same level will likely store the same content, while due to the filtering effect, higher level caches will store different content. Assuming that caches on different levels store completely disjoint content, as for a tree of depth 3 there are 4 cache levels (including the root CCN cache before content the originator), with back of the envelope calculation we may expect to achieve a cache hit of $4C/(|F_{99}|F) = 0.7$.

However, file size plays an important role as well. To understand what changes from an object-wise to a chunk-wise caching, we perform the following experiments. Keeping the ratio $\frac{C}{|F|F} = 10^{-5}$ constant, we let the file and cache size jointly scale, so that also the $C/F = 10^3$ ratio remains

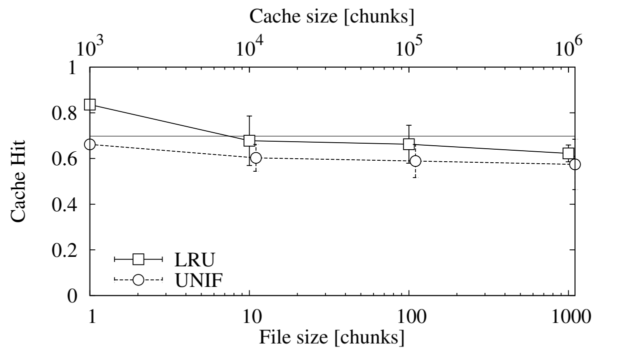
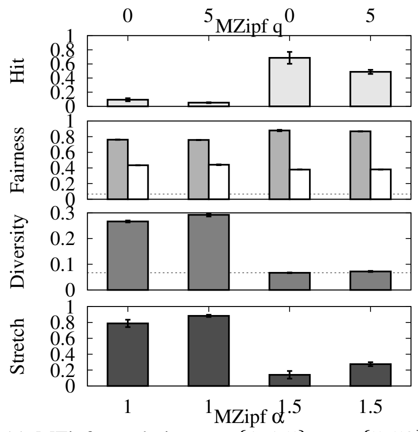


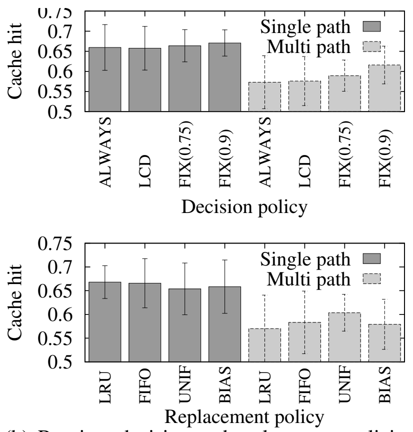
Fig. 3. Cache hit for LRU and FIX( )replacement policies as a function of the file size in chunks.

constant. Results are shown in Fig. 3, as a function of the cache size (YouTube case corresponds to the rightmost scale). First, notice that despite both $\frac{C}{|F|F}$ and C/F are now constant (unlike in Fig. 2) there is a further hidden factor affecting CCN performance. This factor can be identified in the rate at which new object-level requests are issued, which is constant in Fig. 3 across all file sizes. With a constant $W = 1$ interest window, at most one outstanding chunk request can be sent by any user prior to receive the corresponding data and issuing another interest. The download time of the full object is thus related to the network RTT (on the tree, 6 ms up to the root for 1 ms link delay), so that downloading 10^3 chunks long object takes at most 6 sec (reduced in the case of cache hits). Hence, the larger the files, the longer now becomes the transfer of a full object: when a higher number of ongoing downloads compete for the cache space, this in turn lowers the cache hit probability. Second, notice that when files are 1-chunk long, there is a significant difference between LRU and UNIF replacement policies: however, this difference shrinks as file size (and contention) increases, and eventually almost vanishes.

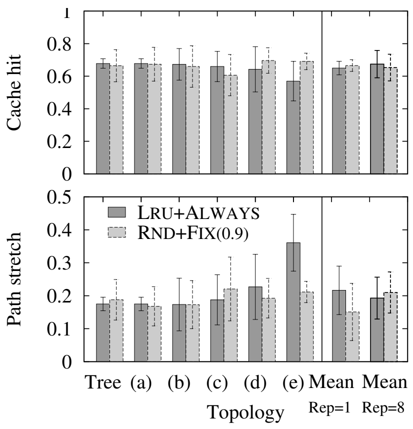
Overall system performance is reported in Fig. 4(a) for varying popularity models: as expected, low α yield to poor system performance, that further worsen for increasing q. It can be seen that performance metrics are highly correlated: when $\alpha = 1$, the closest cache does not necessarily have the chunk of interest, hence a larger stretch. Moreover, as the size of the popular part of the catalog is large, so is the diversity.



(a) MZipf popularity $\alpha \in \{1, 1.5\}$, $q \in \{5, 50\}$



(b) Routing, decision and replacement policies



(c) Topology and multiple repositories

Fig. 4. CCN performance at a glance: popularity, chaching policies, routing, multiple repositories and topology impact.

As far as load fairness is concerned, we report both data (gray bars) and interest (white bars) fairness: notice that as requests for popular content are filtered from the first caches, these are more loaded than upstream ones (explaining the lower interest fairness). Conversely, for highly skewed popularity $\alpha = 1.5$ the same content replicates everywhere, so diversity of the caches approaches the minimum (dotted line), and stretch lowers as well, since the content is often found in close caches.

C. Routing, caching policies and topologies

Considering the YouTube like scenario with $(F, |F|, C, \alpha, q) = (10^3, 10^8, 10^6, 1.5, 0)$ we perform a set of simulations for all combination of 6 topologies, 2 routing strategies, 4 caching decision and 4 replacement policies, averaging over 10 simulation runs. In order for simulations to be comparable across topologies, we still consider a single repository (as in the tree), placed behind a node selected at random. On each topology, 8 nodes (as in the tree) perform object requests at a 1 Hz rate, with a window of $W = 1$ chunk.

We also consider the impact of a single (Rep=1) vs multiple repositories with disjoint content (Rep=8), mimiking in the latter case an AS with several peering links toward different content providers. However, as the impact of multiple repositories is minimal, we report the single repository case to avoid cluttering the pictures, unless otherwise stated.

We first inspect the impact of routing and caching policies in Fig. 4(b). For each routing strategy and all network topologies, we report the average cache hit conditioning over a given decision policy, averaging over all replacement policies (top plot) or conditioning over a replacement policy (bottom plot). Consider single path case first: since we consider a single repository, this case corresponds to a non-regular tree, where the heterogeneity of link propagation delays further shapes the interest (and chunk) arrival process at the different caches. Rather surprisingly, the performance difference across replacement and caching policies is minimal.

Consider now the difference between single vs multiple paths. In case interest travels along multiple paths in parallel, on the one hand this increases the likelihood that an interest packet hits a cache containing the corresponding chunk. At the same time, this also possibly induces a "pollution" of caches along the alternate path, as now the data traveling back will cause eviction on multiple caches. As Fig. 4(b) clearly shows, cache pollution offsets the advantage of reaching a higher number of caches, worsening the overall system performance. Notice that this happens consistently for all decision and replacement policies, with FIX(0.9) and UNIF performing slightly better than the others in the multi-path case.

We then select the most commonly used cache configuration (ALWAYS+LRU) and the best performing one in case of multipath (FIX(0.9)+UNIF) and analyze the impact of the different topologies. Fig. 4(c) reports cache hit and path stretch statistics for (i) tree, (ii) the topologies of Tab. IV and (iii) the average over all topologies. Notice that, on average ALWAYS+LRU and FIX(0.9)+UNIF performance are hardly distinguishable (and rather ALWAYS+LRU is not always the best choice overe all topologies), which holds true for both single and multiple repositories.

V. Discussion and conclusions

This work presents a simulation study of Content Centric Networks (CCN), with special emphasis on caching performance. We consider several aspects including: sizing of catalog and caches, popularity, caching replacement and decision policies, topologies, single vs multiple-path routing strategies. Summarizing our main results, we gather that (i) the impact of the topology is limited, (ii) multi-path routing may play against CCN efficiency, (iii) simple randomized policies perform almost as well as more complex ones, (iv) catalog and popularity settings play by the way the most crucial role.

As for (i), this is not surprising, as [37] that suggests request stream aggregation, more than network topology, to be a key factor in cache placement; (ii) is instead more surprising, and call for future investigation on different routing strategies. Then, (iii) on the one hand confirms the importance of correlated arrivals, that void the IR assumption; on the other hand, this is a positive finding, as it also suggests that simple policies are able to provide good enough performance in CCN. Yet, the most disturbing aspects is that, (iv) unless a proper sizing of the catalog, and assessment of the popularity of its elements, is performed, no definitive answer of CCN efficiency can be claimed – which call for more *systematic* measurement work beyond [24]–[28].

In the following, we aim at extending this work in a number of directions. While our choice of caching policies is already fairly representative, it is however not exhaustive. As far as the decision policies are concerned, we point out that also *deterministic* (e.g., based on hashing of the content as done in [38] for P2P VoD) or *probabilistic distance-based* policies (e.g., as done in CONIC [39]) may be worth investigating. The same goes for *heterogeneous replacement policies*, that were shown to provide moderate improvement in hierarchies of Web caches [40], [41]. In CCN, a promising direction could also be to investigate the *heterogeneous cache size* vs cache placement tradeoff: on the one hand, larger caches should be placed at nodes with higher betweenness centrality (i.e., that lay on a large number of shortest paths); on the other hand, this may not always be feasible due CCN line of speed operation and technological constraints (e.g., large memories are generally also slow and thus may be available only at the edge of the network). Along a similar line, it may be worth jointly investigating routing strategies and cache replacement (e.g., similarly to [18] where object requests are forwarded to the most recent direction the same object was previously forwarded). Finally, the impact of multiple repositories with shared content, user behavior (e.g., [42] points out that 60% of YouTube videos are watched for less than 20% of their duration) and of transport dynamics (e.g., having heterogeneous or time-varying interest window W) needs to be assessed as well.

Acknowledgements

This work was funded by the French ANR project Connect.

References

[1] A. Odlyzko, “Content is not king,” *First Monday*, vol. 6, no. 2, 2001.
[2] J. Choi, J. Han, E. Cho, T. T. Kwon, and Y. Choi, “A Survey on Content-Oriented Networking for Efficient Content Delivery,” *IEEE Communications Magazine*, pp. 121–127, 2011.
[3] V. Jacobson, D. K. Smetters, N. H. Briggs, J. D. Thornton, M. F. Plass, and R. L. Braynard, “Networking Named Content,” in *ACM CoNEXT*, 2009.
[4] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, “Redundancy in network traffic: findings and implications,” in *ACM SIGMETRICS*, 2009, pp. 37–48.
[5] A. Finamore, M. Mellia, M. Meo, M. Munafo, and D. Rossi, “Experiences of internet traffic monitoring with tstat,” *IEEE Network*, 2011.
[6] ccnSim homepage. http://www.infres.enst.fr/~drossi/ccnSim.
[7] S. Podlipnig and L. B. Osz, “A Survey of Web Cache Replacement Strategies,” *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.
[8] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, “Modelling and Evaluation of CCN-Caching Trees,” *IFIP Networking*, 2011.
[9] K. Katsaros, G. Xylomenos, and G. C. Polyzos, “MultiCache : An overlay architecture for information-centric networking,” *Computer Networks*, pp. 1–11, 2011.
[10] S. Arianfar and P. Nikander, “Packet-level Caching for Information-centric Networking,” in *ACM SIGCOMM, ReArch Workshop*, 2010.
[11] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, “Modeling Data Transfer in Content-Centric Networking,” in *ITC*, 2011.
[12] G. Carofiglio, M. Gallo, and L. Muscariello, “Bandwidth and Storage Sharing Performance in Information Centric Networking,” in *ACM SIGCOMM, ICN Worskhop*, 2011, pp. 1–6.
[13] K. Park and V. Pai, “Scale and performance in the coblitz large-file distribution service,” in *USENIX NSDI*, 2006.
[14] N. Laoutaris, H. Che, and I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis,” *Performance Evaluation*, vol. 63, no. 7, 2006.
[15] H. Che, Y. Tung, and Z. Wang, “Hierarchical web caching systems: Modeling, design and experimental results,” *IEEE Journal on Selected Areas in Communications,*, vol. 20, no. 7, pp. 1305–1314, 2002.
[16] T. Wong, G. Ganger, and J. Wilkes, “My cache or yours? making storage more exclusive,” in *USENIX Annual Technical Conference*, 2002.
[17] N. Laoutaris, S. Syntila, and I. Stavrakakis, “Meta Algorithms for Hierarchical Web Caches,” in *IEEE ICPCC*, 2004.
[18] E. J. Rosensweig and J. Kurose, “Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks,” *IEEE INFOCOM*, 2009.
[19] K. Psounis and B. Prabhakar, “A randomized Web-cache replacement scheme,” *IEEE INFOCOM*, pp. 1407–1415, 2001.
[20] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, “On the scale and performance of cooperative Web proxy caching,” in *ACM SOSP*, 1999, pp. 16–31.
[21] M. Fiore, F. Mininni, C. Casetti, and C.-F. Chiasserini, “To cache or not to cache?” in *IEEE INFOCOM*, 2009, pp. 235 –243.
[22] E. J. Rosensweig, J. Kurose, and D. Towsley, “Approximate Models for General Cache Networks,” *IEEE INFOCOM*, pp. 1–9, 2010.
[23] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler, “Small is not always beautiful,” in *USENIX IPTPS*, 2008.
[24] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon, “I tube, you tube, everybody tubes: analyzing the world’s largest user generated content video system,” in *ACM IMC*, 2007, pp. 1–14.
[25] C. Zhang, P. Dhungel, and K. Di Wu, “Unraveling the BitTorrent ecosystem,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 1164–1177, 2010.
[26] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “Youtube traffic characterization: a view from the edge,” in *ACM IMC*, 2007, pp. 15–28.
[27] A. Bellissimo, B. Levine, and P. Shenoy, “Exploring the use of bittorrent as the basis for a large trace repository,” *Tech. Rep.*, 2004.
[28] M. Hefeeda and O. Saleh, “Traffic modeling and proportional partial caching for peer-to-peer systems,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, 2008.
[29] M. Busari and C. Williamson, “ProWGen: a synthetic workload generation tool for simulation evaluation of Web proxy caches,” *Computer Networks*, vol. 38, no. 6, pp. 779–794, 2002.
[30] N. Spring, R. Mahajan, and D. Wetherall, “Measuring isp topologies with rocketfuel,” in *ACM SIGCOMM*, 2002.
[31] R. Ogier, V. Rutenburg, and N. Shacham, “Distributed algorithms for computing shortest pairs of disjoint paths,” *IEEE Transactions on Information Theory,*, vol. 39, no. 2, 1993.
[32] Ccnx homepage. http://www.ccnx.org/.
[33] Y. Zhou, T. Fu, and D. Chiu, “Statistical modeling and analysis of p2p replication to support vod service,” in *IEEE INFOCOM*, 2001.
[34] V. Alfred, J. Peter, and D. Jeffrey, “Principles of optimal page replacement,” *Journal of the ACM*, vol. 18, no. 1, pp. 80–93, 1971.
[35] M. Mitzenmacher, A. Richa, and R. Sitaraman, “The power of two random choices: A survey of techniques and results,” *Handbook of randomized computing*, vol. 1, pp. 255–312, 2001.
[36] Omnet++ homepage. http://www.omnetpp.org/.
[37] R. Fonseca, M. Crovella, and B. Abrahao, “On the Intrinsic Locality Properties of Web Reference Streams,” in *IEEE INFOCOM*, 2003.
[38] Y. Liu and G. Simon, “Peer-assisted time-shifted streaming systems: Design and promises,” in *IEEE ICC*, 2009.
[39] Y. Zhu, M. Chen, and A. Nakao, “Conic: Content-oriented network with indexed caching,” in *IEEE INFOCOM Workshops*, 2010, pp. 1–6.
[40] M. Busari and C. Williamson, “Simulation evaluation of a heterogeneous web proxy caching hierarchy,” in *IEEE MASCOT*, 2001, pp. 379–388.
[41] H. Che, Z. Wang, and Y. Tung, “Analysis and design of hierarchical web caching systems,” in *IEEE INFOCOM*, 2001, pp. 1416–1424.
[42] A. Finamore, M. Mellia, M. Munafo, R. Torres, and S. Rao, “Youtube everywhere: Impact of device and infrastructure synergies on user experience,” in *Tech. Rep.*, 2011.