Evaluating CCN multi-path interest forwarding strategies

Giuseppe Rossini[a], Dario Rossi[a]

[a]*Telecom ParisTech, 46 rue Barrault, 75634 Paris, France*

Abstract

This work addresses the performance evaluation of Content Centric Networks (CCN). Focusing on a realistic YouTube-like catalog, we conduct a thorough simulation study of the main system performance, focusing on multi-path interest forwarding strategies but considering several other ingredients such as network topology, content popularity, caching decisions and replacement policies.

Summarizing our main results,(i) catalog and popularity settings plays by far the most crucial role (ii) the impact of the strategy layer comes next, with naive forwarding strategies playing against CCN efficiency, (iii) simple randomized caching policies perform almost as well as more complex ones, (iv) the impact of the topology is limited. 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.

1. Introduction

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 threatens 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 [11]) have started proposing caching of objects (or object chunks) as network primitives. While these proposals 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) [18] is one of 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

Email addresses: `dario.rossi@enst.fr` (Giuseppe Rossini), `giuseppe.rossini@enst.fr` (Dario Rossi)

de facto replace traditional buffers, so to avoid redundant network traffic [3]: 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. 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[12].

The contribution of this paper is as follows. First, we develop `ccnSim` an efficient and scalable chunk-level CCN simulator, that we make available to the scientific community as open source software [1]. `ccnSim` allows to assess CCN performance in scenarios up to four order of magnitude larger with respect to the those considered in the current literature. For instance, in this work we simulate large CCN cache sizes (10^6 chunks), catalog sizes (10^8 files), file sizes (10^3 chunks) and topologies (up to 68 nodes). Second, we take care in defining a realistic scenario, investigating the effect of wide parameter ranges that, along with the simulator software, can hopefully assist and promote cross-comparison in the research community. Third, we conduct a thorough simulation campaign, considering several popularity settings, 6 topologies, 4+4 cache decision+replacement policies, 1 single-path +12 multi-path forwarding strategies. Results of this campaign allow us to conclude that (i) catalog and popularity settings plays by far the most crucial role (ii) the impact of the strategy layer comes next, with naive forwarding strategies playing against CCN efficiency – though we find strategies with similar caching efficiency of shortest path forwarding, and added benefits of multi-path resilience and reduced repository load, (iii) simple randomized caching policies may be good enough, (iv) provided that heterogeneous latencies are taken into account the impact of the network topology is limited.

2. Related work

Despite a large caching literature (see e.g., [25] 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. In the following we merely provide a brief overview of the most relevant related work [4, 5, 6, 7, 10, 11, 13, 14, 19, 20, 21, 26, 28, 29, 32, 33], referring the reader to [15] for a more comprehensive comparison of the above work.

A first limit of current caching work is that, with few exceptions[26, 19, 4, 7, 6, 11], *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, e.g., BitTorrent in P2P networks, that lately was brought into CDN[24] as well.

Concerning caching policies, as pointed out in [20], much attention has been devoted to *replacement policies* issues (over 40 policies are overviewed in[25]), while *decision policies* have being studied sporadically (to the best of our knowledge, only by [9, 31, 20]). As far as *replacement policies* are concerned, Least Recently Used (LRU) has been used in the context of CCN [26, 7, 6] and of the more general ICN context [28, 11, 19]. Only few work considers other policies, such as Most Recently Used (MRU) and Most Frequently Used (MFU) in ICN [19]. Still, due to the fact that CCN caching operations must happen at line speed, most of the existing decision and replacement policies are not of practical relevance, as [4] points out. Thus, 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[4]. As far as *decision policies* are concerned, generally the assumption is to Leave a Copy Everywhere (LCE), i.e., new content gets always cached. Few exceptions to this rule come from the Web [9, 31, 20] or CCN[4] contexts: however, the approach in [9] doesn't apply to CCN due to implementation complexity, while DEMOTE [31] is known to poorly perform on network of caches. With this respect, only the Leave Copy Down (LCD) policy [20] is simple enough to be worth implementing in CCN. Again, an even simpler policy is considered by [4], 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 [30] for Web, and [13] for ad hoc domain) would likely violate CCN line of speed constraint.

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 both analytical and simulation based studies. First, most of previously developed models rely 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[7, 6], though the analysis is limited to simple cascade or tree topologies. Conversely, though the approximate cache model in [29] 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 [11, 19] (employing synthetic topologies generated with GT-ITM), most simulative work still considers simple topologies (e.g., cascade or trees[6, 26, 7, 4]) and are thus not suitable to investigate multi-path issue, which is in the genes of CCN[18]. 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.,

Table 1: System parameters investigated in this work

Parameter		Meaning	Values
CCN	c	Chunk size	10 KBytes
	C	Cache size	10^6 chunks (10 GB)
Catalog	$\|F\|$	File number	10^8 files
	F	File size	10^4 chunks (10 MB, geom. distributed)
	$\|F\|F$	Catalog size	10^{15} bytes (1 PB)
	R	Number of repository	{1,2}
Popularity	α	Zipf exponent	{1,1.5}
	q	Mandelbrot-Zipf plateau	{0,5}
Caching		Cache replacement	{FIFO,LRU,UNIF,BIAS}
		Cache decision	{LCE,FIX(P),LCD}
Strategy-Layer		Path Policy	{Single-Path, Multi-Path, Multi-Repository}
		Path Selection	{Uniform, Round-Robin, Parallel}
		Path Retention	{None, Argmin}
Network		Topology	Tree, Abilene, Geant, Tiger2, DTelecom, Level3

add disks). In CCN instead, due to the fact that interest and data packets need to be serviced at the Network Interface Card (NIC) speed, cache size is technologically limited by memory access speed[4]. As it is not possible to arbitrarily increase size of caches on board of CCN routers, it becomes thus 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.

3. System and scenario description

While for reason of space, we invite the reader to [18] 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)*. 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 and travels along the PIT information (flow balance). Whenever an interest is satisfied, the corresponding entry in the PIT is then removed. Conversely, when a new interest reaches a CCN router that already has a PIT entry for the same data, the interest is not propagated, but the interface from which the interest is received is appended to the PIT (interest aggregation).

We summarize in Tab. 1 the parameter space and support we investigate. On the one hand, among the most important factors affecting CCN performance, we individuate cache and catalog size (Sec. 3.1) and content popularity (Sec. 3.2). Yet, these factors are constrained by technological limits (e.g., cache size) or determined by the environment (e.g., catalog size and popularity). In this work, a great care is taken to evaluate CCN in realistic operational points (as, e.g., one cannot simply increase the cache size to ameliorate CCN performance).

On the other hand, overall system performance also depends on cache-related or mutli-path mechanisms that are specific to CCN: the aim of this work is to broadly explore their design space. Indeed, on different *network topologies*, interest may be forwarded along either a single shortest path, or through multiple-paths, depending on the *strategy layer* (Sec. 3.3). Then, once data travels back in the CCN data plane following PIT "crumbles", CCN routers along the path participate in a caching process, that can be modeled as being composed by two distinct *caching policies* (Sec. 3.4). First, a *decision policy* 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* in case the cache is full. In the remainder of this section, we motivate our decisions pertaining the boundaries of the space summarized in Tab. 1.

3.1. Cache and catalog size

First, CCN chunks are expected to be packet-size (1KB[26]-10KB[6, 7]), hence much smaller w.r.t other ICN architectures (16MB[19]). In our work, we select 10KB chunks as in [6, 7]: despite a CCN chunk size is not specified in [18], we believe that in reason of the CCN overhead –due to interest packets and large headers for content naming and security issues– chunk sizes smaller than 10KB would be an overkill.

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. As far as CCN router cache size is concerned, [4] starts from the observation [3] that about half of the caching benefits at packet level happen in the first 10 sec.: considering a capacity of 1Gbps, [4] sizes to about 12 GB the amount of memory that can be addressed at line speed (employing a two-levels address scheme). Yet, the cache size typically considered in simulation is much smaller (from 6.4MB [26] to 50MB [7] and 2GB [6]), 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.

The number of files in the considered catalogs can be as low as 250 objects[29], topping to 20K[6, 7] objects for the largest ones. Taking into account also the object size (in chunks) considered in those work, largest catalogs varies between 2Mchunks[7] and 13.8Mchunks [6], 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. We consider a realistic Internet catalog, namely the YouTube portal, due to its wide popularity and growth of video content. Given the approximate number of videos (about 10^8[8]) and their size (geometrically distributed with average 10MB [16]), the catalog size is on the order of PB (i.e., 10^{15} Bytes).

We expect system performance to be determined by the ratio of the cache C over the catalog size $|F|F$. We point out that this ratio varies in the ICN literature between a minimum of 0.25%[7]-0.5%[19] to a maximum of 10%[29]-20%[19]. Yet, considering realistic CCN cache and Internet catalog sizes, 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.

3.2. *Content Popularity*

Another important aspect that concur in determining the system performance is the content popularity model. Rather typically [28, 6, 7, 20], ICN 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 [19] resorts to a Mandelbrot-Zipf distribution (that is also determined by the plateau q), while [28] also includes uniform popularity, and [6, 7] consider several classes distributed with Zipf popularity (with uniformly popular objects within each class).

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[19]-2.5[6]. However, let us evaluate the rank $|F_{99}|$ of the 99-th percentile of the Zipf distribution: intuitively, $|F_{99}|$ represents the number of objects in the catalog that make up the 99% bulk of users' requests. For high values of $\alpha \geq 2$[7, 6] the 99-th rank converges to the same value *irrespectively of the catalog size*: 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 expect CCN performance to be drastically determined by the popularity exponent α. While a broad exploration of α, q settings is available in [15], here we limitedly focus on $\alpha \in \{1, 1.5\}$ (that are about the centerfold of the parameters explored in the literature) and $q \in \{0, 5\}$ (as per [17]).

3.3. *Multi-path strategy layer*

To promote cross comparison we resort to real publicly available network topologies, that are listed in Tab. 1, and that have a size varying between 11 and 68 nodes (for lack of space, we invite the reader to [15] for more details about the topologies). As a reference, we also consider a standard binary tree topology with 15 nodes and 8 leafs. In the following, we assume backbone links have infinite capacity (or, otherwise stated, to operate at a load level well below congestion).

Irrespectively of the considered topology, a peculiar aspect of CCN is that named content can in principle take *any* path in the network. In CCN, decisions where to send (or replicate) interests are taken by the *strategy layer*. In this work, we assume that an external routing process (and name resolution scheme) provides to the strategy layer *multi-path alternatives* from the requester to the content originator. For the sake of simplicity, in this work we limitedly consider at most two alternative paths. The strategy layer has then to decide how to use these paths, on a chunk-by-chunk basis.

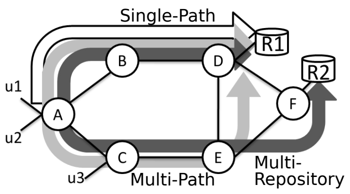

Figure 1: Toy-case example

We examplify the situation with a toy-case network, where we consider decisions taken at node A to reach content located in possibly multiple repositories R_1, R_2. On the one hand, (i) in case node A serves requests of user u_1 employing only the shortest-path (single-path, white arrow) to the closest repository R_1 (that pass through B, D), it may miss previous requests of user u_3 (stored at C, E). On the other hand, if strategy layer exploits more than a single path, it can either (ii) employ multiple alternative[1] paths to the same repository (multi-path, light-gray arrow), or (iii) employ multiple shortest paths to different repositories (multi-repository, dark-gray arrow). These *multi-path policies* tradeoff between (i) the fact that shortest path routing cannot fully exploit the benefit of in-network caching, as caching happens only en-route toward the repository, potentially missing closer cached copies; (ii) the fact that alternate repositories may be faraway, so that the interest may travel longer paths; (iii) the fact that path diversity toward the closest repository may be poor, so that few alternative caches are visitied beyond those visited by shortest path routing.

In both cases, the strategy layer faces further decisions concerning the *multi-path selection* process, such as (i) exploring both paths in parallel, or (ii) alternating between paths (and in this case, if uniformly at random, or deterministically as in round robin). The tradeoff is in this case between (i) an increased interest load, that violates the flow-balance and possibly causes cache contention and (ii) a possibly slower convergence to the repository.

Finally, the selection process can happen either (i) every chunk or (ii) once per object (keeping thus a per-object state about the path). We denote this choice as *multi-path retention*. In case (i), decisions are taken for every chunk, so no path is retained. Conversely, in case (ii) an interest[2] is sent over all the possible paths at time 0, and the path retained is the one along which the first data will come. The path retention tradeoffs (i) simplicity for (ii) greater efficiency, as it may be preferable to forward interest *toward the closest cached copy* (as opposite to the closest repository).

3.4. Cache Decision and Replacement Policies

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. However, at the network scale, systematically caching all chunks in all nodes may translate in a low cache diversity, hence reducing CCN efficiency. Rather, as already noticed in [4] and [20] 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 several decision policies that try to achieve an implicit distributed coordination between caches: (i) LCE, new content is always cached; (ii) FIX(P), caching decisions are taken uniformly at random as in [4], with a fixed probability $P \in \{0.75, 0.9\}$; (iii) LCD, on each miss, the Leave a Copy Down policy [20] moves the content a single hop down, but downstream router further away do not cache the request.

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

[1]As primary path, we consider the shortest path between the requester and the originator. As secondary path, we instead use [23] to find the shortest path that is the most diverse from the primary

[2]In case parallel path are used, the same interest is expressed over multiple paths; otherwise, a different chunk is requested over each alternative path.

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 replace either (i) LRU, the least recently used chunk; (ii) FIFO, the oldest inserted chunk; (iii) UNIF, a chunk selected uniformly at random; (iv) BIAS, the most popular between two chunks selected at random (detailed description in [15]).

We consider LRU as it is the default caching choice, FIFO and UNIF as simpler replacement. Finally, we design a new, counter intuitive, cache replacement policy that selects chunks to be replaced *proportionally to their popularity* (BIAS). 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. This policy leverages on the "power of two" principle [22], which was already used in the context of Web caching [27].

4. Simulation results

We implement a chunk-level CCN simulation model that univocally address content chunks, faithfully representing the CS, PIT and FIB data structures and CCN operations such as interest aggregation. However, aiming at scalability and efficiency, we do not implement the full details of CCN naming and security. To promote cross-comparison in the scientific community, we make the simulator, written in C++ under the Omnet++[2] framework, available as open-source software at [1].

We conduct a thorough simulation campaign, consisting of more than 10,000 simulations exploring over 1,000 individual system parameter settings. In this section, we report the most interesting results obtained from the campaign: due to space limitation, our aim is not to provide an exhaustive coverage of our results, but rather to convey a few relevant messages to the scientific community, in the most compact way. For the interested reader, an extended set of results in our extended technical report[15].

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 standard deviation bars). By default, we consider a single YouTube-like repository served by CCN routers, to which aggregates of clients are attached, issuing 10 requests per second. CCN routers implement LRU+LCEcaching policy and their strategy-layer forward interests along the shortest-path toward the repository.

Caching performance is usually expressed in terms of the *cache hit* probability. In CCN networks, additional metrics are needed in order to capture the network-wide perspective. As user centric-metric, beyond the usual *cache hit* probability, we consider the *distance* in terms of the number of CCN backbone hops d that the data chunk has

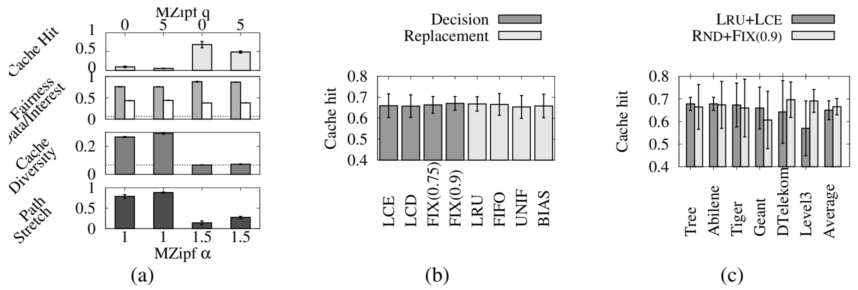

Figure 2: CCN performance at a glance: (a) popularity, (b) caching policies and (c) topology impact.

actually traveled in the network (that correlates with user delay). Finally, as network-wide metric, we consider the *interest load*, i.e., the number of links that interests have crossed before hitting a cache. Interest load correlates with CCN router cost (as interest processing must be done at line speed) and with network load (as data is generated in reply to interests, and will flow backtracking the PIT).

4.1. *Performance at a glance*

Considering a YouTube-like scenario $(F, |F|, C) = (10^3, 10^8, 10^6)$, we report overall system performance in Fig. 2(a) for varying popularity skews $\alpha \in \{1, 1.5\}$. As expected, low α yields 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 data travels a longer distance. This in turns lowers the cache hit rate of individual CCN content stores, and translate into a higher interest cost as well. For highly skewed popularity $\alpha = 1.5$ the same content replicates everywhere, so that content of interest is often found in closer caches, hence with a reduced interest cost. We see that α may undermine whether CCN will be able to deliver the promised breakthrough: in case of $\alpha \approx 1$, it seems that a technology change (i.e., the speed and size of memories for CCN content stores) is evidently needed to increase the cache over catalog ratio, and so CCN performance. In the meanwhile, a refined estimation of object popularity may be necessary to evaluate CCN fit to serve different Internet catalogs (e.g., YouTube, BitTorrent, Netflix, etc.).

4.2. *Caching policies and topologies*

We now inspect the impact of caching policies in Fig. 2(b) considering $(\alpha, q) = (1.5, 0)$. We point out that the choice of such a skwewed content popularity is not ment for an absolute assessment of CCN performance, but rather for a relative assessment of the impact of several parameters, and is thus perfectly justified.

For each routing strategy and all network topologies, we report the average cache hit (i) conditioning over a given decision policy, averaging over all replacement policies (dark-gray bars) or (ii) conditioning over a replacement policy (light-gray bars). Consider single path routing: with 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. Yet, rather surprisingly, the performance difference across cache replacement and decision policies is minimal.

In reason of wire-speed constraint of CCN, it could be thus worth investigating other, simpler, combination than the most commonly used LCE+LRUpair. Hence, we assess the impact of different topologies considering also random decision and replacement strategies FIX(0.9)+UNIF. Fig. 2(c) compares cache hit on (i) a binary tree, (ii) the 6 topologies of Tab. 1, and (iii) the average over all topologies. Notice that, average LCE+LRU vs FIX(0.9)+UNIF performance are hardly distinguishable: furthermore, LCE+LRU is *not* always the best choice overe all topologies.

4.3. Strategy layer

As we have seen that caching policies and topologies plays a limited role in determining system performance, we now fix the LCE+LRUcaching policies and limitedly consider the Geant network to assess the impact of the *strategy layer*. We further consider the case where all links have homogeneous propagation delay, and the heterogeneous case with real delays reflecting the interconnection lengths.

We compare several multi-path alternative to the single shortest path case. Specifically, we consider either (i) alternatives paths to the same repository, or (ii) shortest paths toward multiple repository. We further consider that the strategy layer can either (i) alternate between paths either randomly or round-robin or (ii) use both paths in parallel. Finally, we consider the case where (i) after an initial probing phase using multiple paths for the first chunk of a given object, the first reply path is retained and used for subsequent interests of that object or (ii) multiple paths are used for all interests of an object. Overall, we test 12 multi-path strategy layers, but we report only the most interesting combinations in the following.

Fig. 3-(a) reports the average download distance as a function of the node distance from the closest repository. Notice that filled points refer to scenario with heterogeneous delay, empty ones to homogeneous delay. Two lines report reference for cases where the content is entirely downloaded from the closest repository, or is downloaded with shortest path routing. Fig. 3-(b) reports instead the average interest load, i.e., the number of links that each interest has crossed before hitting a cache. Light-gray color is used for heterogeneous delay, dark-gray for homogeneous delay. Three lines report the average distance from the closest repository $E[sp(x, R1)]$, the shortest path distance from the alternate repository $E[sp(x, R2)]$, the length of the alternate path toward the closest repository $E[ap(x, R1)]$.

As expected, when multiple paths are used, these are longer than the distance to the closer repository, hence the download distance increases. Looking at Fig. 3-(a), this effect is especially noticeable only for nodes that are close to the repository, with a peak at two hops; conversely, faraway nodes will likely find the content cached somewhere before the repository, with peformance closer to shortest path routing. Similarly, when multiple paths are used in parallel, the interest load roughly doubles. Considering Fig. 3-(b), path retention may however succesfully mitigate the interest load increase, at a price of keeping per-object state in CCN routers.

We point out that in case paths are used in an *alternate* fashion, there is no difference between a uniform randomized criterion and a deterministic round-robin one (hence, we do not show this in the plots). Intuitively, if deterministic decisions are not coordinated between nodes, they are equivalent to random decisions from the overall network perspective. When instead multiple paths are used in a *parallel* fashion, this

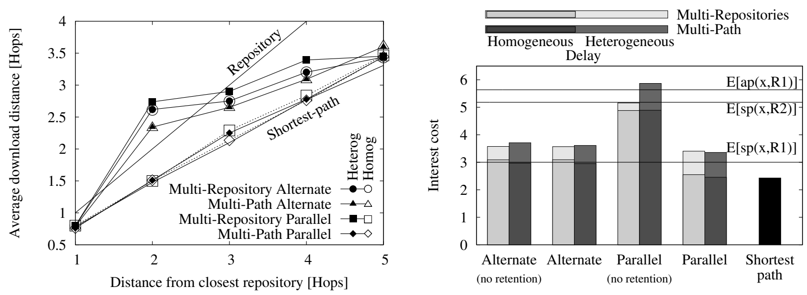

Figure 3: Impact of strategy layer on average distance from a cache, and interest/data cost

lead to a more aggressive probing policy, that has higher chances to find closer cached content.

Also, we see that multiple alternative path *toward the same repository* should be preferred to the use of multiple shortest paths toward several repositories (furthermore this lead to more robust choices in case of heterogeneous delay). Intuitively, this is because CCN values diverse paths as they explore different caches, more than because they ultimately lead to different repositories. Interestingly, this happens despite the average alternate path toward the closest repository in on average longer than the shortest path toward the alternate repository, i.e., $E[ap(x, R1)] > E[sp(x, R2)]$ in Fig. 3-(b).

The above observation has an important consequence. First, this means that, unless the ultimate aim is to reduce load at the repositories, information concerning a single repository suffices. Furthermore, in CCN efficiency is maximum when the closest cached copies is found: since cache update dynamics are much faster than routing dynamics possibly, this information will likely not be available in the FIB. Hence, there may no need for the routing protocol to disseminate information about optimate alternate paths to the repository either, since an opportunistic exploration of a CCN node neighborhood may suffice.

Overall, we find that naïve-multi path strategies exploiting other paths than the shortest one may lead to explore a longer distance – increasing the overall network load, and reducing the cache hit rates. At the same time, we recognize that multi-path benefits come from an increased resilience (in case shortest path fail) and to a possibly reduced load at the repository (whose uplink bandwidth can be a scarce resource with respect to router capacities). In this case, it seems that multi-path strategies that complement single-path routing with an opportunistic exploration of a CCN node neighborhood may be worth exploring

5. Discussion and conclusions

This work presents a simulation study of Content Centric Networks (CCN) caching performance. We consider several aspects including: sizing of catalog and caches, popularity, caching replacement and decision policies, topologies, and epecially focus on single vs multiple-path interest forwarding strategies. Summarizing our main results, we gather that (i) catalog and popularity settings play a crucial role (ii) the impact of the strategy layer comes next, with naïve multi-path forwarding strategies playing against CCN efficiency, (iii) simple randomized policies perform almost as well as more complex ones, (iv) the impact of the topology is limited, provided that heterogeneous latencies are accounted for.

While (i) is an expected results, this also means that much remains to be done in order to fully assess whether CCN can deliver the promised breakthrough. The lack of common evaluation scenario is a critical point that the ICN community should address in the short term: due to its tremendous impact, this call for *broad and systematic* measurement work beyond [8, 16, 17] to gather reliable popularity models.

Then, (ii) strategy layer has the second largest impact after catalog settings. Specifically, we gather than naïve multi-path routing strategies may play against CCN caching efficiency, for which a simpler shortest path interest forwarding may be preferable. Yet, multi-path benefits may come from other aspects, orthogonal to caching efficiency, such as increased resilience and reduced repository load. With this respect, we find the strategy layer should probe multiple paths in parallel for the first few chunks of every object, forward then the interests along the path leading to the closest cached copy. This strategy indeed achieves similar caching efficiency of the shortest path routing, with the addeed benefits of multi-path.

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.

Finally, (iv) is not surprising, as [14] that suggests request stream aggregation, more than network topology, to be a key factor in cache placement. This also means that, provided that latency heterogenity is properly accounted for, the network topology is a less important detail with respect to the previous scenario parармeters.

We believe that interest forwarding and routing should deserve further work. One direction concerns the interaction of routing and fowarding: while this work assumes the availability of multiple-paths in the FIB used by the strategy layer, there is currently no routing procotocol that disseminates this name-based FIB information: hence, the strategy layer may not always have rich multi-path information at its disposal. Furthermore, our analysis show that benefits of multi-path follows from the exploration of multiple closeby caches: hence, the strategy layer may not even need such multipath information. As a consequence, alternative strategies, up to the extreme case of opportunistic strategies assuming no FIB information at all, should be assessed as well.

Another direction includes the interplay between interest forwarding with the caching of the data on the return path. A promising direction is thus the *joint design* of forwarding strategies and cache replacement policies (e.g., similarly to [28] where object requests are forwarded to the most recent direction the same object was previously forwarded, and thus likely cached). Such joint design may include (simple) replacement strategies based on *deterministic* [21] or *probabilistic distance-based* [33] decisions, to increase the diversity of CCN caches. Finally, it may be worth investigating if *heterogeneous replacement policies* may provide some moderate performance improvement for CCN as they did for the Web[5, 10]. Both directions are part of our current research interest.

Acknowledgements

This work was funded by the French ANR project Connect.

[1] ccnSim homepage. `http://www.infres.enst.fr/~drossi/ccnSim`.

[2] Omnet++ homepage. `http://www.omnetpp.org/`.

[3] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee. Redundancy in network traffic: findings and implications. In *ACM SIGMETRICS*, pages 37–48, 2009.

[4] S. Arianfar and P. Nikander. Packet-level Caching for Information-centric Networking. In *ACM SIGCOMM, ReArch Workshop*, 2010.

[5] M. Busari and C. Williamson. Simulation evaluation of a heterogeneous web proxy caching hierarchy. In *IEEE MASCOT*, pages 379–388, 2001.

[6] G. Carofiglio, M. Gallo, and L. Muscariello. Bandwidth and Storage Sharing Performance in Information Centric Networking. In *ACM SIGCOMM, ICN Worskhop*, pages 1–6, 2011.

[7] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. Modeling Data Transfer in Content-Centric Networking. In *ITC*, 2011.

[8] M. Cha, H. Kwak, P. Rodriguez, Y.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.

[9] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE JSAC*, 20(7):1305–1314, 2002.

[10] H. Che, Z. Wang, and Y. Tung. Analysis and design of hierarchical web caching systems. In *IEEE INFOCOM*, pages 1416–1424, 2001.

[11] 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*, pages 121–127, 2011.

[12] A. Finamore, M. Mellia, M. Meo, M. Munafo, and D. Rossi. Experiences of internet traffic monitoring with tstat. *IEEE Network*, 2011.

[13] M. Fiore, F. Mininni, C. Casetti, and C.-F. Chiasserini. To cache or not to cache? In *IEEE INFOCOM*, pages 235 –243, 2009.

[14] Rodrigo Fonseca, Mark Crovella, and Bruno Abrahao. On the Intrinsic Locality Properties of Web Reference Streams. In *IEEE INFOCOM*, 2003.

[15] D. Rossi G. Rossini. Caching performance of content centric networksunder multi-path routing (and more). Technical report, Telecom ParisTech, 2011.

[16] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *ACM IMC*, pages 15–28, 2007.

[17] M. Hefeeda and O. Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Transactions on Networking*, 16(6), 2008.

[18] 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.

[19] K. Katsaros, G. Xylomenos, and G. Polyzos. MultiCache : An overlay architecture for information-centric networking. *Computer Networks*, pages 1–11, 2011.

[20] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis. Meta Algorithms for Hierarchical Web Caches. In *IEEE ICPCC*, 2004.

[21] Y. Liu and G. Simon. Peer-assisted time-shifted streaming systems: Design and promises. In *IEEE ICC*, 2009.

[22] M. Mitzenmacher, A. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. *Handbook of randomized computing*, 1:255–312, 2001.

[23] R.G. Ogier, V. Rutenburg, and N. Shacham. Distributed algorithms for computing shortest pairs of disjoint paths. *IEEE Trans. Inf. Theory,*, 39(2), 1993.

[24] K.S. Park and V.S. Pai. Scale and performance in the coblitz large-file distribution service. In *USENIX NSDI*, 2006.

[25] S. Podlipnig and L. B Osz. A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys*, 35(4):374–398, 2003.

[26] I. Psaras, R. G Clegg, R. Landa, W. K. Chai, and G. Pavlou. Modelling and Evaluation of CCN-Caching Trees. *IFIP Networking*, 2011.

[27] K. Psounis and B. Prabhakar. A randomized Web-cache replacement scheme. *IEEE INFOCOM*, pages 1407–1415, 2001.

[28] E. J. Rosensweig and J. Kurose. Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks. *IEEE INFOCOM*, 2009.

[29] E. J. Rosensweig, J. Kurose, and D. Towsley. Approximate Models for General Cache Networks. *IEEE INFOCOM*, pages 1–9, 2010.

[30] 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*, pages 16–31, 1999.

[31] T.M. Wong, G.R. Ganger, and J. Wilkes. My cache or yours? making storage more exclusive. In *USENIX Annual Technical Conference*, 2002.

[32] Y. Zhou, T.Z.J. Fu, and D.M. Chiu. Statistical modeling and analysis of p2p replication to support vod service. In *IEEE INFOCOM*, 2001.

[33] Y. Zhu, M. Chen, and A. Nakao. Conic: Content-oriented network with indexed caching. In *IEEE INFOCOM Workshops*, pages 1–6, 2010.