Large scale simulation of CCN networks

Giuseppe Rossini, Dario Rossi

Telecom ParisTech

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 play 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.

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*s, rather than merely being provided with *connectivity* among two addressable entities, a number of architectures (overviewed in [CHC+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) [JSB+09] is one of the most promising. Yet, the potential impact of such that architecture 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. 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[‡]. 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), file sizes (up to 10^3 chunks) and topologies (up to 68 nodes). Briefly speaking we were able to simulate up to four order of magnitude more compared to the current scenarios considered in the literature. Second, we conduct a thorough simulation campaign, considering several popularity settings, 6 topologies, 2 routing strategies, 4 cache decision and 4 cache replacement policies.

[†] This work was carried out at LINCS http://www.lincs.fr and funded by ANR Connect http:://www.anr-connect.org

[‡] For reason of space... just google for it :)

Tab. 1: System parameters investigated in this work

Parameter	Meaning	Values
c	Chunk size	10 KBytes
F	File size	up to 10^4 chunks (10 MB, geometrically distributed)
$\|F\|$	Number of files	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
Network	Network topology	{Abilene, Geant, Tiger2, DTelecom, Level3}

2 System and scenario description

While for reason of space we invite the reader to [JSB+09] 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 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, several other aspects may affect CCN performance: e.g., on different *network topologies* (having between 22 and 68 nodes), interest may be forwarded along either a single shortest path or through multiple-paths, depending on the *routing algorithm* (Sec. 3.2). 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* establishes whether or not to cache the current data. The set of decision policies taken into account holds subsequent elements: *Always* (caches every chunk it receives), *Fix* (caches with fixed probability) and *Leave Copy Down* (LCD, caches only if the immediate downstream node for the specific chunk).

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. Eviction algorithms are more studied within the literature. In this paper, we consider: the *FIFO* policy (evicts the first arrived chunk), *LRU* (evicts the least recently used), *Uniform* (evicts random), and *Biased* (chooses two random elements and evicts the most popular). Finally, a critical point consist in the selection of a the request popularity model, that we model as a Zipf-Mandelbrot law (the probability $p(i)$ that a content i is going to be requested by a client is $p(i) = C/(i+q)^\alpha, \quad C = \sum_{i=1}^{|F|} 1/(i+q)^\alpha$), on whose precise settings the community has not however agreed upon yet. For the sake of readability, we summarize in Tab. 1 the explored parameter space.

3 Simulation results

`ccnSim` implements a chunk-level CCN simulation model that univocally addresses content chunks, faithfully representing the CS, PIT, and FIB data structures and CCN operations such as interest aggregation. The simulator, written under the Omnet framework, has been designed to be *extremely scalable*: to promote cross-comparison in the scientific community, we make the simulator available as open-source software.

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,

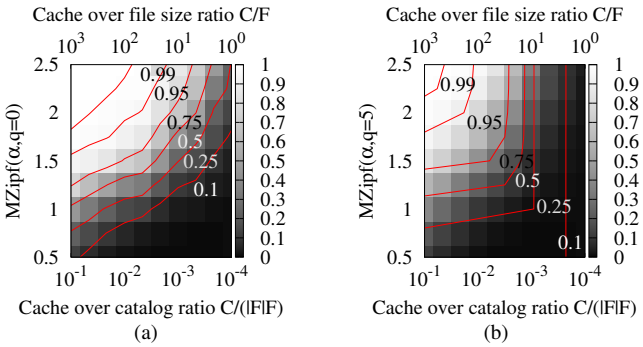

Fig. 1: Contour plot of cache hit for catalog/cache ratio vs MZipf α when q is 0 (a) and 5 (b).

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 a companion technical report [GR11].

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).

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, and to further qualify the cache diversity vs link usage tradeoff. As user centric-metric, we consider the *path stretch* $d/|P|$ as the number of CCN backbone hops d that data chunks travel 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 topologies.

3.1 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 α, q settings of MZipf popularity. The root of the tree is connected to the unique repository, while the 8 leafs perform object-level requests with exponentially distributed arrival times at a 1 Hz rate. We limitedly consider standard replacement and decision policies (LRU, ALWAYS) and 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. Performance is also determined by the average number of files that can be stored in a single cache C/F (reported on the top x-axis, varying from 1000 to 1).

We depict the contour plot of the cache hit probability for the above settings in Fig. 3, for $\alpha \in [0.5, 2.5]$ and $q \in \{0, 5\}$. Notice that the explored parameter range covers almost the full support of the cache hit metric. For small values of $\alpha \leq 1$, caching is not effective when the catalog size is large. For large values of $\alpha \geq 1.5$, as the number of files that can be stored in the cache exceeds the 99% request percentile, this leads to an overly simple caching problem (that the plateau q may temper as we see in comparing the left and right contours). Hence, the promised disruption may not really happen depending on the real value of (α, q): further research is needed to accurately estimate this critical parameter.

3.2 Routing, caching policies and topologies

We now consider a realistic YouTube-like scenario $(F, |F|, C, \alpha, q) = (10^3, 10^8, 10^6, 1.5, 0)$ and perform a set of simulations for all combination of 6 topologies, 2 routing strategies, 4 caching decision and 4 replacement policies (192 settings, averaging over 10 simulation runs per setting; parameter selection is thoroughly

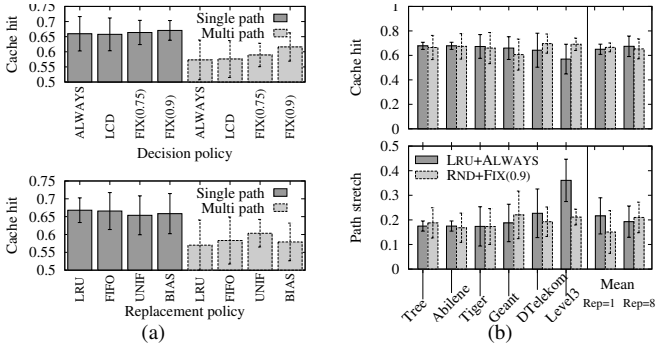

Fig. 2: Impact of topologies, routing, caching decision and replacement policies.

motivated in [GR11]). For simulations to be comparable across topologies, users are attached to 8 CCN border routers in all topologies (as in the tree), and perform object requests with exponential arrivals, with a window of $W = 1$ chunk. We consider either a single repository $Rep = 1$ (as in the tree), placed behind a node selected at random, or that the content is distributed over $Rep = 8$ random repositories (as the impact of multiple repositories is minimal, we report the single repository case unless otherwise stated).

We first inspect the impact of routing and caching policies in Fig. 2(a). Over all topologies, for each routing strategy we report the average cache hit conditioning over a given decision policy (i.e., averaging over all replacement policies, top plot) or conditioning over a replacement policy (bottom plot). Consider the 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 decision 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. 2(a) 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 ALWAYS+LRU (the most common configuration) and FIX(0.9)+UNIF (as it exhibits the best performing in case of multipath), and analyze the impact of topology in Fig. 2(b). Interestingly, we gather that the cache hit of these two policies combinations differ of only about 4% on average. Furthermore ALWAYS+LRU is not always the best choice over all topologies, and this holds true for both single and multiple repositories settings. Still, we point out that the impact of the topology is again modest, with a spread between the worst and best cache hits of about 10% (for either policies combination).

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 far the most crucial role above all.

References

[CHC+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.

[GR11] D. Rossi G. Rossini. Caching performance of content centric networksunder multi-path routing (and more). Technical report, Telecom ParisTech, 2011.

[JSB+09] Van Jacobson, Diana K Smetters, Nicholas H Briggs, James D Thornton, Michael F Plass, and Rebecca L Braynard. Networking Named Content. In *ACM CoNEXT*, 2009.