A dive into the caching performance of Content Centric Networking

Giuseppe Rossini, Dario Rossi
Telecom ParisTech, Paris, France

first.last@enst.fr

***Abstract*—Content Centric Networking (CCN) is a promising architecture for the diffusion of popular content over the Internet. While CCN system design is sound, gathering a reliable estimate of its performance in the current Internet is challenging, due to the large scale and to the lack of agreement in some critical elements of the evaluation scenario.**

In this work, we add a number of important pieces to the CCN puzzle by means of a chunk-level simulator that we make available to the scientific community as open source software. First, we pay special attention to the locality of the user request process, as it may be determined by user interest or language barrier. Second, we consider the existence of possibly multiple repositories for the same content, as in the current Internet, along with different CCN interest forwarding policies, exploiting either a single or multiple repositories in parallel.

To widen the relevance of our findings, we consider multiple topologies, content popularity settings, caching replacement policies and CCN forwarding strategies. Summarizing our main result, though the use of multiple content repositories can be beneficial from the user point of view, it may however counter part of the benefits if the CCN strategy layer implements naïve interest forwarding policies.

I. Introduction

With the mass-market adoption of bandwidth intensive applications, the explosion of video over the Internet and its forecasted growth, the largest part of network traffic is today represented by *popular content* that is downloaded repeatedly across network links from several users [2].

To relieve traffic redundancy in the current content-oblivious Internet, *service-specific* optimizations such CDN caches and P2P overlays, have generally been built *over the top* of the infrastructure. This trend has recently reversed with the proposal of a number of content-oriented architectures, referred to as Information Centric Networking (ICN) [6], that advocate service-independent in-network caching as a fundamental lower-level building block of the future Internet.

The move from a network offering access to hosts to a network offering access to data constitutes, beyond any doubt, a paradigm shift. In the relatively short history of telecommunication networks, this shift may be comparable the introduction of packet switching, who ultimately led to the Internet as we know it. However, as any important shift, the move toward ICN networks will happen only if it can provide a substantial economic advantage (along with a viable business model) by means of a technological breakthrough – which we aim at assessing in this paper.

Content Centric Networking (CCN) [8] is one of such proposals: in CCN, content is sent in reply to *interest packets* addressing *named data chunks*, any may get cached by any CCN router along the way back to the request originator. In CCN routers, caches de facto replace traditional buffers: while the aim of an IP router is to dismiss packets as quickly as possible (i.e., by forwarding them to the next hop interface), the aim of a CCN router is instead to keep the most popular data for the longest possible (useful) time.

Yet, despite the large caching literature that already exists, a number of architectural details make CCN study challenging and novel at the same time. Differently from previous studies, CCN routers are arranged as arbitrary network topologies. Moreover, CCN stores very small data chunks, of the order of a single packet, as opposite to traditional architectures where full objects (or very large chunks) are generally cached. Furthermore, each new object request generates a stream of requests for data chunks: hence, the request arrival process is highly correlated, unlike in traditional caching studies. Finally, the large size of CCN router caches and the tremendous size of Internet catalogs, makes the study of CCN performance a daunting task.

In this work, we dive into CCN performance by means of simulation, angling for a realistic Internet-wide YouTube-like catalog. By exploring several system aspects –such as multiple topologies, user locality and content popularity models, CCN forwarding and caching replacement strategies– we get a grip of CCN performance. We offer our simulation tool to the scientific community as open source software at [1].

II. Reference CCN system model

In CCN, users request content by sending *interest packets* for named data chunks to their CCN access router: in case such data is already stored in the router cache (also known as *content store* in CCN), it is immediately sent back in reply to the user interest. Otherwise, the CCN router inserts a reference to the interface from where the interest came from in the *Pending Interest Table* (PIT) and forwards the interest packet on some interfaces (aka *faces* in CCN) according to a decision taken by a *strategy layer* on *Forwarding Information Base* (FIB) data disseminated by a *routing protocol*. In case multiple requests for the same content hit a CCN router that has already created a PIT entry (but has not received the content yet), the PIT is appended with references to the latter requests:

this way, CCN performs *request aggregation*, reducing further the network traffic. Potentially, in case the data of interest is not cached at any content store of intermediate CCN routers, interest packets reach the original data repository. The data chunk sent in response to the interest packet travels then back according to PIT information, and PIT references are removed once interests are satisfied by sending data on the corresponding interfaces.

Despite the *strategy layer* plays an important role in the interest forwarding, in the original CCN proposal the "default strategy is to send an interest on all broadcast capable faces then, if there is no response, to try all the other faces in sequence" [8]. While this strategy is efficient is searching the local neighborhood and will eventually find the content of interest, it may be complex to tune in practice. Indeed, while TCP relies on the estimation of Round Trip Time (RTT) beteween data and acknowledgements for self-clocking, CCN breaks the end-to-end assumption: as the CCN termination end-point may vary from chunk to chunk, the same goes for the corresponding RTTs between interests and data. Setting an appropriate face timeout is thus not trivial, since long values may result in high delay for the user, while short values may trigger unnecessary requests. Also, the strategy is not optimized in case data is available at multiple source repositories – which is commonplace in today's Clouds, due to data replication over multiple points of presence (PoP) for resiliency, load balancing and low service latency.

In this work, we consider the impact that alternative strategy layers have on the overall CCN caching performance. As we describe in the reminder of this section, to gather representative CCN performance we consider a network of CCN routers, configured with different cache replacement policies, and interconnected by either regular or real topologies. The CCN network serves a realistic YouTube-like catalog, whose objects may have a geographically biased popularity and are possibly stored at multiple PoPs.

A. Evaluation scenario

While work on ICN/CCN performance has started to appear [3], [4], [6], [9], it generally considers (i) simple cascade or trees topologies (with the exception of [6]), (ii) modest catalog sizes (varying from 1,000 [6] to 20,000 objects [4]) and (iii) small cache sizes (varying from 6.4MB [9] to 2GB [4]).

To counter these limitations we (i) consider both a regular 4×4 torus topology, as well as the topology of the GEANT network, interconnecting the European research and education institutions. As for (ii), we consider the catalog size of YouTube, estimated by [5] to amount to 10^8 video files, having geometrically distributed size with 10 MB average [7], yielding to a 1 PB catalog. As for (iii), motivated by technological constraints of nowadays memory access speeds, we select as in [3] the size of individual caches equal to 10 GB: thus, each cache can only store a very small portion (10GB/1PB=10^{-5}) of the whole catalog.

We assume to operate in a non-congested regime and consider links of infinite bandwidth, so that the network delay matches the propagation delay of each link. We notice further that CCN studies typically implement a Least Recently Used (LRU) cache replacement policy [4], [6], [9]. Yet, we argue as in [3] that, due to the fact that CCN caching operations must happen at line speed, even LRU may be complex to implement in practice – hence, we also consider a random replacement policy (RND) in the evaluation. Finally, we set CCN data chunk size to 10KB as in [4].

B. Multiple repositories and strategy layers

As it happens for IP networks, we assume that forwarding and routing happen at two separate timescales. In other words, an external routing process modifies the FIB at low frequency, so that paths between CCN routers and repositories can thus be considered constant during each simulation.

As previously stated, nowadays Cloud services may replicate data in several PoPs. An high level view of the scenario is depicted in Fig. 1 (using the 4×4 torus topology for the sake of illustration). We consider that content is either stored at a *single repository* $|R| = 1$ or at *multiple repositories* $|R| \geq 1$. In the single repository case, we forward interests along the shortest path toward the repository. In the multiple repositories case, we instead consider two forwarding policies: specifically, interests are forwarded either (i) along the shortest path leading to the *closest repository*, or (ii) along multiple shortest paths leading to *all repositories* in parallel. Since we do not consider local access networks, no broadcast interface is available, as routers are interconnected with point-to-point links. Hence, performance of the original strategy [8] should lay between the above two extreme cases. Also, we expect the use of parallel paths to improve user-centric performance, at the price of an increased interest (and data) load on CCN routers.

For the sake of simplicity, in the following we consider that content can be stored in either a single or four data-center locations $|R| = \{1, 4\}$. On each simulation, the location of the $|R|$ repositories is randomized, and results are averaged over 40 runs to smooth out stochastic variability.

C. Content popularity and user locality

As usually assumed in the literature and pointed out in [5] for YouTube, we consider *content popularity* to follow a Zipf distribution. At the same time, as there is no consensus on the exponent α of the Zipf distribution in the CCN literature (α varies between 1 [6] and 2.5 [4]), we select a large range of values for $\alpha \in [0.5, 2.5]$ for a more robust analysis (see Sec. III-A).

We model the aggregate arrival of new *object requests* as a Poisson process (having average rate $\lambda = 20Hz$). Notice that once a user requests the first chunk of a given object, it will also generate the subsequent interests for the following chunks to completely download the file. Notice further that, while in IP networks a user sending rate is typically determined by some end-to-end congestion control algorithm (e.g., TCP), this end-to-end model no longer holds under CCN. At the same time, as CCN does not implement a TCP replacement for the time being, we assume for the sake of simplicity that users

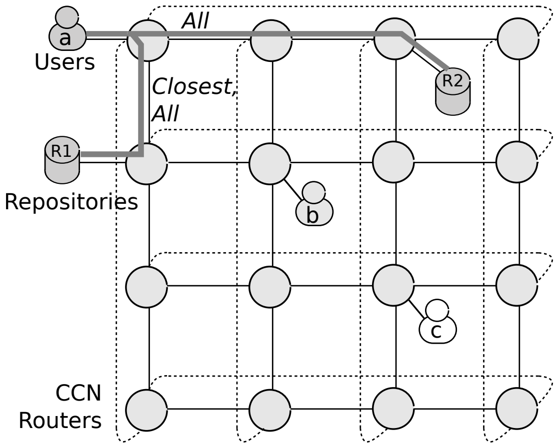

Fig. 1. High level view of the CCN scenario.

have a fixed $w = 1$ interest window (i.e., users wait for their outstanding interest to be satisfied by a data chunk before sending out a new interest).

Concerning the *user requests*, we foresee that irrespectively of the content popularity model and user arrival rate, there can be a locality bias in the request generation process. Reasons for this bias are easily figured out by considering, e.g., the language barrier in the GEANT network, that interconnects several European countries each speaking a different idiom.

To take this bias into account without affecting the overall popularity distribution we "polarize" the user requests as follows. For each new request, we first randomly generate an object ID according to a Zipf popularity distribution. Then, we map this new request for ID to a random user u in the network (say, user a in Fig. 1), attached to an access CCN router $C(u)$: in case this is the first request for ID, we set $C_0(ID) = C(u)$. Once subsequent requests for ID are generated, we bias the user extraction process so to favor the selection of users closer to $C_0(ID)$ (say, user b will be more likely chosen than c in Fig. 1). Specifically, a candidate user v is extracted at random over the whole network, and the distance $d(C(v), C_0(ID))$ is evaluated: the request will then be mapped to this new user with a probability $P(d)$ that monotonously decreases with this distance (represented with fading user color in Fig. 1). If the mapping fails due to $P(d)$, new candidates are extracted at random until the mapping succeeds. In this work, for the sake of simplicity, we select

$$P(d) = \max\left(0, 1 - \frac{d}{\mathcal{L}+1}\right) \quad (1)$$

with a locality radius $\mathcal{L}$ within which the content is confined equal to $\mathcal{L} = 2$. In practice, candidates behind the same access CCN router $C_0(ID)$ will always be accepted, candidates at one (two) CCN hop(s) will be accepted with probability 2/3 (1/3), and faraway candidates will never be accepted. This allows for some "penetration" in the language barriers, as sometimes neighboring countries have shared languages (e.g., France, Belgium and Luxembourg, or Austria, Germany and Switzerland, or Italy and Switzerland, etc.).

At the same time, language barriers may not apply in the case of a national ISP: notice that (1) degenerates in the uniform model for $\mathcal{L} \to \infty$, so so that requests may come *uniformly* at random from any user.

III. Performance Evaluation

This section reports results of our simulation campaign. We use a custom CCN chunk-level simulator, developed under the Omnet++ framework, that we make available to the scientific community at [1]. For each simulation point, we collect the metrics of interest when the cache hit rate converges to a stationary value (which usually take about 1 hr of simulated time from the time the caches fill up), and we average results over 40 simulation runs.

As for any caching system, CCN promises to reduce the amount of traffic flowing in the network, and to reduce latency for the end users as well. Typically, the efficiency of a cache is measured in terms of the *hit rate* or, in CCN terms, the probability that the content chunk of interest is found at a given content store.

As CCN implements a network of caches, it is also important to consider the *hop distance* traveled by an interest before a content store is hit – and the corresponding data travels back until the interest originator. Indeed, the hop distance roughly reflects the overall load on the network and the end-user delay. At the same time, from the chunk-level hop distance it is hard to relatively compare performance across different topologies (due to difference in the absolute path lengths) and users (as it mixes chunks of popular and unpopular content). We thus define a normalized distance, that relatively weights the hop distance with respect to topological properties of the considered network. Given a user u and an object j, we denote the downloading *effort*:

$$\eta_{u,j} = \frac{\sum_{i=1}^{\text{size}(j)} d_i}{|D|\text{size}(j)} \quad (2)$$

where size(j) represents the size (in chunks) of object j, d_i represents the distance traveled by the $i-th$ chunk of file j and $|D| = d(C(u), R(j))$ represents the shortest path distance between the CCN access router $C(u)$ of user u and the nearest repository $R(j)$ storing j. Notice that d_i only counts CCN router hops, so that η varies in the range $[0, 1]$: when $\eta \to 0$ a user finds the whole object in the content store of its CCN access router; when $\eta \to 1$, the content is downloaded entirely from the nearest repository.

Finally, as a simple metric to evaluate the whole system efficiency, we take into account the raw *number of data and interest messages* sent over the CCN network (during the first hour of simulated time, as simulation duration may differ across runs).

A. The popularity dilemma

Due to lack of consensus on the object popularity settings, let us start by a preliminary investigation of the system performance for varying exponent α of a Zipf distribution. For

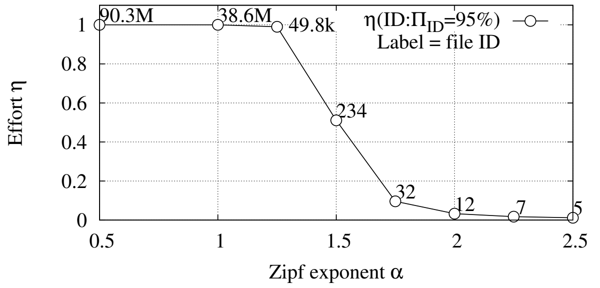

Fig. 2. Effort values for the set of files representing the 95-th percentile of the request distribution. Labels report the ID of the file corresponding to the 95-th request percentile.

the sake of simplicity, we refer to a *baseline scenario* with a 4×4 torus network, LRU replacement policy, uniform demands and a single repository. By definition, the Zipf distribution probability equals $P(X = i) = \frac{1/i^\alpha}{C}$ with $C = \sum_{j=1}^{|F|} 1/j^\alpha$ where i is the rank of the i-th most popular file in a catalog of size $|F|$ files. Thus, $P(X = i)$ corresponds to the percentage of requests for the i-th object, and we further denote $\Pi_i = \sum_{j=1}^{i} P(X = j)$ as the cumulative percentage of requests directed to the set of i most popular objects.

Fig. 2 reports, as a function of α, the object representing the 95-th percentile of the request distribution, averaged over all users (i.e., $\mathbb{E}_u[\eta_{u,ID}]$ with ID such that $\Pi_{ID} = 95\%$); notice that this ID varies with α, and is reported on the labels next to the curve. For low values of α (e.g., $\alpha = 0.5$), the 95% of requests span over a almost the entire catalog (e.g., 90.3M out of 100M videos): in turn, this yields to a continuous stream of interests reaching the repository ($\eta \approx 1$). As alpha grows, the number of objects that make the 95% bulk of requests is significantly smaller, making caching highly effective. For large α however, the problem becomes trivial: e.g., at $\alpha = 2$, only 120MB of cache are required to cache the 12 files (out of the 100M files catalog) responsible for 95% of the request, so that $\eta \approx 0$. The sharp transition phase happens between $\alpha = 1.25$ (the bulk of requests concerns about 50k files, or 500TB storage space) and $\alpha = 1.5$ (234 files or about 2.4GB), that we therefore consider as boundaries for the Zipf exponent in the remainder of the evaluation.

B. Network-centric performance

Fig. 3 shows CCN performance at a glance, reporting the chunk-level hit rate (top plots), the raw amount of data chunks flowing in the network (middle plots) and the effort to download the file corresponding to the 95%-th request percentile ($\eta_{95\%}$ for short, bottom plots). For each popularity setting ($\alpha = 1.5$ left, $\alpha = 1.25$ right) labels in the plot report the performance of the *baseline* scenario with LRU caches, single repository and uniform requests. Bars report the absolute difference with respect to the baseline, gathered for different caching replacement (LRU vs RND), number of repositories and strategy layer settings (*single* path for $|R| = 1$, and *closest* vs *all* repositories when $|R| = 4$), and request locality (uniform $\mathcal{L} \to \infty$ vs biased $\mathcal{L} = 2$).

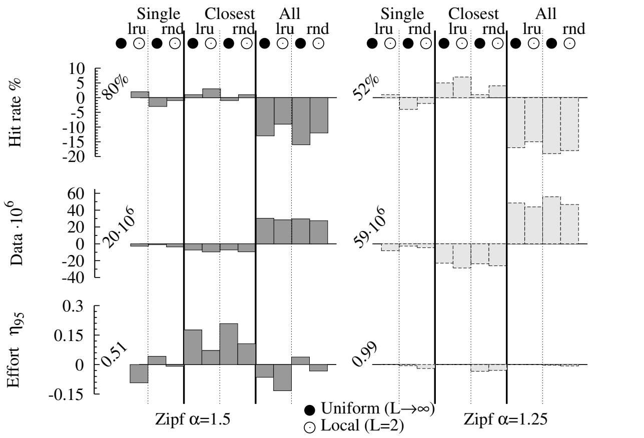

Fig. 3. CCN performance at a glance: hit rate (top), raw data amount (middle) and download effort (bottom). Labels report baseline performance for LRU policy, under uniform requests directed to a single repository. Bars show performance differences with respect to baseline.

As expected, absolute CCN performance varies significantly with α: e.g., hit rate achieves 80% when $\alpha = 1.5$ (52% at $\alpha = 1.25$), and the 95% most popular requests only travel halfaway $\eta_{95\%} = 0.51$ toward the repository (is practically downloaded from the repository $\eta_{95\%} = 0.99$). At the same time, the *relative* trends across scenarios are consistent across popularity settings: notice that when a simple RND replacement is used instead of a LRU strategy, the hit rate reduces by only a few percentage points; similarly, localizing content requests only mildly ameliorate performance.

Above all, the strategy layer settings have the largest impact. Consider first the case where interests are forwarded over the shortest path (labeled as *single* for $|R| = 1$, *closest* for $|R| = 4$). When multiple repositories are available, nodes have on average a closer repository (w.r.t the single repository case), hence exhibit a higher hit rate, and as a consequence the amount of data reduces significantly. At the same time, notice that the download effort (for the ID corresponding to 95% of the cumulative requests) increases: despite paths are now shorter, multiple independent diffusion trees rooted at each repository now forms, which *reduces the likelihood of interest aggregation at each CCN router*.

When the strategy layer forwards interests toward all repositories in parallel, this clearly translates into a higher data volume: yet, notice that the number of data packets only slightly more than doubles, despite $|R| = 4$ paths are used in parallel. Moreover, though the per-object download effort decreases (which is due to the more aggressive strategy, and beneficial from the user point of view), this happens to the detriment of the per-chunk hit rate. This behavior is explained through the *increased cache contention, as the use of parallel paths forces eviction on multiple CCN content stores*.

C. User-centric performance

We now turn the attention to the performance that users may expect when downloading objects corresponding to different levels of popularity. Fig. 4 depicts the per-object effort η_{ID}

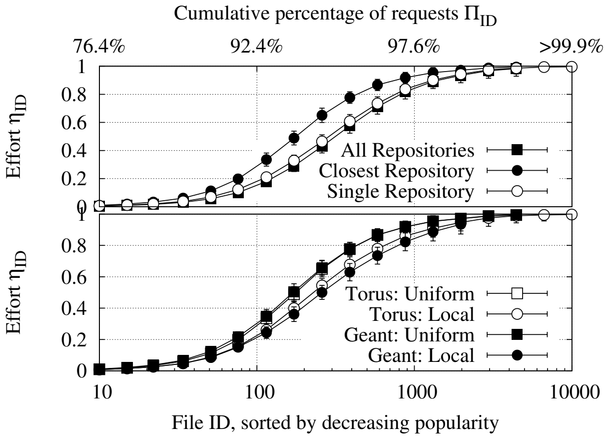

Fig. 4. Per-object performance: impact of strategy layer (top), topology and request locality (bottom) on the download effort.

where ID represents the popularity rank, with lowest ID objects being the most popular (considering $\alpha = 1.5$ for the sake of brevity). The top plot of Fig. 4 shows the joint impact of the strategy layer and of the repository availability. We report object whose ID falls in the range $ID \in [10, 10^4]$, as this limited support already spans the full range for the $\eta_{ID} \in [0, 1]$ metric. Irrespectively of CCN settings, the top-10 objects gets always cached at the access CCN router ($\eta_{ID} = 0$ for $ID \leq 10$), while objects toward the tail of the catalog are always served by the repository ($\eta_{ID} = 1$ for $ID \geq 10^4$). Performance of $ID \in [10, 10^4]$ (corrisponding to roughly 1/4 of the requests) are instead affected by the strategy layer.

Interestingly, the use of a single path in case of multiple repositories (closest policy) actually increases the relative effort (by about a factor of two for the first 100 elements). As previously observed, this is due to multiple independent CCN trees building up. Despite the *absolute* path length is shorter than in the single repository case (recall from Fig. 3 that the absolute number of messages reduces under the closest policy), the interests travel *relatively* further than before, due to the reduced level of interest aggregation in the PIT. When instead all the shortest paths to all repositories are used in parallel, the download effort reduces further compared to both the closest and single policies – though the gain remains modest due to the increased cache competition.

Finally, the impact of the network topology (torus vs GEANT) and of the request spatial properties (uniform vs locally biased requests) is shown in the bottom plot of Fig. 4. For the sake of illustration, we fix other settings to LRU caching policies, popularity exponent $\alpha = 1.5$, with $|R| = 4$ repositories served by a closest policy. First, notice that performance is very similar under either topology, and that furthermore the simple regular torus topology provides conservative results (for the effort metric). As expected, spatially confined requests are beneficial to CCN: intuitively, non-uniform requests reduce caching contention and increase aggregation, yielding to a higher caching efficiency.

IV. Conclusions

This paper assesses CCN efficiency in serving a YouTube-like catalog by means of simulation. First, we show CCN performance to be strongly dependent on the popularity model of users requests, as even slight variation on the Zipf exponent α can have a dramatic impact on the system performance. A precise answer to whether CCN can keep its promise is thus strongly related to this crucial scenario aspect.

Second, though the locality of user requests can simplify the caching problem, it does however not suffice in making the problem *scalable* – hence, the ability to build *fast* (multi Gbps) and *large* (multi GB or TB) caches seems another necessary piece to solve the CCN puzzle. Third, on the positive side, our results show that a random replacement policy can be used as a replacement for LRU with limited performance loss – simplifying thus line speed requirement. Fourth, topology of the CCN network seems to have the least impact, so that investigation of simple topologies (e.g., regular torus) may represent a good first approximation.

Fifth, the strategy layer has, by far, the largest impact on CCN performance. Our results show that, in case the original content is stored by multiple repositories, part of CCN appeal may vanish if simple forwarding strategies are used. Indeed, in case CCN forwards requests for data that is not locally available toward the closest repository only, this reduces the level of request aggregation at CCN routers, loosing one of the main CCN advantages. In case CCN forwards requests toward all repositories in parallel, this instead yields to an increased competition in the CCN router caches, resulting in the eviction of cached content from multiple paths, and reducing the overall cache hit probability. As such, the design of more intelligent CCN strategy layers, that either (i) exploit aggregation or at least (ii) reduce the cache contention, remains an open point.

Acknowledgements

This work was carried out at LINCS http://www.lincs.fr and funded by the ANR Project CONNECT

References

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

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

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

[4] 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.

[5] 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*, pages 1–14, 2007.

[6] 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.

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

[8] V. Jacobson, D. Smetters, N. Briggs, J. Thornton, M. Plass, and R. Braynard. Networking Named Content. In *ACM CoNEXT*, 2009.

[9] Ioannis Psaras, Richard G Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. Modelling and Evaluation of CCN-Caching Trees. *IFIP Networking*, 2011.