Framework and Algorithms for Operator-managed Content Caching

Lorenzo Saino, *Member, IEEE,* Ioannis Psaras, *Member, IEEE,* and George Pavlou, *Fellow, IEEE*

***Abstract*—We propose a complete framework targeting operator-driven content caching that can be equally applied to both ISP-operated Content Delivery Networks (CDNs) and future Information-Centric Networks (ICNs). In contrast to previous proposals in this area, our solution leverages operators' control on cache placement and content routing, managing to considerably reduce network operating costs by minimizing the amount of transit traffic and balancing load among available network resources. In addition, our solution provides two key advantages over previous proposals. First, it allows for a simple computation of the optimal cache placement. Second, it provides knobs for operators to fine-tune performance. We validate our design through both analytical modeling and trace-driven simulations and show that our proposed solution achieves on average twice as many cache hits in comparison to previously proposed techniques, without increasing delivery latency. In addition, we show that the proposed framework achieves 19-33% better load balancing across links and caching nodes, being also robust to traffic spikes.**

***Index Terms*—Cache placement; caching; ICN; CDN;**

I. Introduction

INTERNET traffic has been growing exponentially over the past few years mainly fuelled by the dramatic increase of content distribution [1]. This trend has made it necessary for content providers to use geographically distributed caches to serve their content, in order to improve user-perceived latency and throughput as well as to reduce transit costs. As a result, Content Delivery Networks (CDNs) have rapidly grown in terms of both presence and overall traffic carried and are expected to grow steadily for the foreseeable future.

Two key trends are expected to influence content distribution in both the short and long term. The first one, with short term impact, is represented by the rapid growth of Network CDNs, *i.e.,* CDNs managed by network operators (see, for example [2], [3], [4]), which is driven by both technological and commercial incentives. The second one, with longer term impact, is the increasing momentum gained by the emerging Information Centric Networking (ICN) paradigm within both the research and operator/vendor community. This paradigm revisits the Internet architecture targeting content distribution as one of the main use cases. The most prominent ICN architectures, *i.e.,* CCN [5] and NDN [6], envisage the deployment of ubiquitous packet caches in network routers.

Both trends are leading to a scenario in which network operators (*i.e.,* ISPs) will be in charge of operating a networked caching infrastructure, which will likely be very complex, comprising several nodes and carrying a considerable amount of traffic. In a different fashion to traditional CDNs, whose main objective is to improve the performance of content distribution for their customers, network operators also have the additional objective of minimizing the operating cost of their network. This includes reducing the amount of interdomain traffic crossing transit links and ensuring homogeneous utilization of resources by evenly spreading traffic across network links and caching nodes. With its knowledge of topological characteristics, an operator can exert more control on network routing and cache placement, which it can use to better design and operate its distributed caching infrastructure.

The general problem of designing caching systems for content distribution has received considerable attention in the past (see Sec. II for an overview of previous work). However, very little work addressed the specific requirements of network operators (*e.g.,* reduction of network operating costs) while leveraging their strengths (*i.e.,* knowledge of topology and control on routing and cache placement). More importantly, all previously proposed solutions exhibit two fundamental limitations. First, they do not address the problem of how to optimally place caches, which is of fundamental importance for building cost-effective caching infrastructures. Second, their rigid designs give operators limited ability to fine-tune performance according to their requirements.

In this paper we fill this gap by proposing an integrated framework to achieve efficient operator-driven caching that can be equally applied to both Network CDNs and, in the longer term future, to ICNs. Our solution builds upon our previous work in this area [7] and leverages managing hash-routing techniques: the latter have been proposed in the past to route requests among caching nodes co-located in the same physical facility (see, for example [8], [9], [10]) but were never considered for geographically distributed environments. Hash-routing maps each content item to a specific node of a cluster based on a hash function. When a content request reaches a domain node, it is redirected to the responsible caching node by computing the result of a hash function on the URI (or any other unique identifier) of the requested content item.

Overall, our proposed framework provides the following advantages over state-of-the-art caching techniques:

- Greater cache hit ratio, resulting in minimized inter-domain traffic and subsequent transit cost savings.
- Predictable performance, making the caching infrastructure easy to model and optimize.
- Tunable performance in terms of latency, cache hit ratio and robustness to demand variation, giving operators greater flexibility.

Lorenzo Saino is with Fastly, Inc. Ioannis Psaras and George Pavlou are with the Department of Electronic and Electrical Engineering at University College London. Contact addresses: Lorenzo Saino (lsaino@fastly.com), Ioannis Psaras (i.psaras@ucl.ac.uk), George Pavlou (g.pavlou@ucl.ac.uk).

- Inherent load-balancing across links and nodes, robust against localized hotspots and flash crowd events.

We evaluate the performance of our framework with trace-driven simulations using real ISP topologies and real traffic traces. Our evaluation shows that our framework achieves a 2x cache hit ratio in comparison to previously proposed techniques without affecting delivery latency. In addition, it provides a 19-33% better load balancing across network links and caching nodes, which is robust to local traffic spikes and flash crowd events. We make the code required to reproduce these results publicly available in [11].

II. In-network content caching

In this section, we examine the distributed content caching problem and review the main techniques proposed so far, pointing out their shortcomings that our framework addresses.

According to RFC 7929 [12] the in-network caching problem can be partitioned into three well-defined subproblems:

- **Content placement and content-to-cache distribution**, which deals with the problem of deciding which content items to place in which caching node and how to distribute them to those nodes.
- **Request-to-cache routing**, which deals with how to route content requests from a requester to a suitable caching node that holds a copy of the requested content.
- **Cache placement**, which deals with deciding/optimizing the placement and sizing of caching nodes.

Following this classification and given that our framework provides solutions for the above three areas, we discuss the state of the art for each of them in turn. We specifically focus on techniques applicable to the case of operator-controlled caches, which is the use-case investigated in this paper.

A. Content placement and distribution

Content items can be placed in caches in either a *proactive* or a *reactive* manner.

With proactive content placement, caches are pre-populated during off-peak traffic periods. The placement is normally computed by an off-line optimization algorithm on the basis of historical data and/or future predictions and repeated periodically. Many algorithms have been proposed to determine optimized content placement under a variety of objective functions and constraints [13, 14, 15, 16, 17, 18, 19].

Instead, with reactive content placement, content items are stored in caches as a result of cache misses in a read-through manner. Within a caching node, content items can be replaced according to policies such as Least Recently Used (LRU) or First In First Out (FIFO) or more sophisticated policies [20]. If a request traverses multiple caches before hitting the content, the simplest content placement strategy is to leave a copy of the content in every node traversed, which is known as Leave Copy Everywhere (LCE). However, this strategy causes a high degree of redundancy as all caches along the path consume resources to hold identical items. To mitigate this issue, a number of meta algorithms have been proposed to selectively cache each delivered content in a subset of the nodes of the path, such as LCD [21], CL4M [22] and ProbCache [23].

There is consensus that proactive placement is preferable to reactive placement only in the case of workloads characterized by a limited content catalogue and predictable request variations, such as Video on Demand (VoD) [15], [24]. Netflix, the world's largest VoD content provider, uses proactive content placement in their video caching infrastructure [25].

All other types of traffic are characterized by rapid variations of content popularity that would obliterate the gains provided by an optimized proactive placement [16]. Therefore, to the best of our knowledge, all CDNs for Web traffic populate their caches reactively [26], [27], [28]. This is also the placement strategy selected by all ICN architectures proposing ubiquitous packet caches in network routers [5], [6].

In the light of these considerations, we designed our framework to operate based on reactive content placement principles.

B. Request routing

Request routing strategies can be broadly ascribed into two categories: *opportunistic on-path* and *coordinated off-path* routing.

With on-path request routing, content requests are routed over a network of caches towards the content origin using shortest path routing and are served from a cache only if the requested content item is available at a node on that path. This routing strategy is highly scalable as it does not require any coordination among caching nodes. However, it normally suffers reduced cache hits, as any item outside the shortest path will never be hit.

Edge caching is also a (simpler) case of opportunistic on-path routing. In edge caching, requests are routed to the closest cache but in case of a cache miss they are forwarded directly to the content origin. This is for example how Google Global Cache operates [28].

With off-path coordinated routing, requests can be served by a nearby node even if not on the shortest path to the origin. This, however, comes at the cost of higher coordination to exchange content availability information among caches. Off-path routing can be implemented using a centralized or distributed content-to-cache resolution process. In a centralized resolution process, a (logically) centralized entity with a global view of cached contents is queried before routing a content request and it returns the address of the closest node storing the requested content item. This approach is however suitable only for systems operating under proactive content placement. For reactive caching systems with high rate of content replacement (which also include ICN architectures, where items are cached at a chunk-level granularity), a number of more scalable off-path request routing algorithms have been proposed [29], [30].

Request routing design presents a clear tradeoff between scalability and efficiency. The limited scalability of off-path routing particularly limits the availability of design choices for reactive caching systems and ICN architectures, which are of our interest. Our proposed framework addresses this tradeoff by enabling off-path request routing without any coordination, since the content-to-cache mapping is computed in a distributed manner by a hash function. However, this

comes at the cost of traffic detours, introducing longer paths and hence, possibly, greater latency, especially in tail and worst case scenarios, where content items are fetched from origin. Nonetheless, as we show later, with careful cache placement and content routing, it is possible to effectively address the side effects of detours and achieve low average latency while maintaining great scalability and cache hit ratio.

C. Cache placement

Optimal cache placement is dependent on content placement, request routing, the network topology and the content request patterns.

Several studies have investigated the optimal cache placement and sizing with notable results [31], [32], but only for the specific case of proactive content placement, which, as discussed in Sec. II-A, is a suitable approach only for VoD applications.

The problem is considerably more complex in the case of reactive content placement, which instead has much wider applicability. Networks of caches operating under common cache replacement policies, such as LRU and FIFO, are difficult to model. The main source of complexity comes from modeling the behaviour of caches receiving miss streams from other caches [33], [34]. In fact, although there exist analytical models capturing well the behavior of common cache replacement policies in a single cache or tandem configurations [35], [21], [36], extending this analysis to arbitrary network topologies is very hard and methods proposed so far are computationally very expensive and fairly inaccurate [37]. All this makes the optimal cache allocation problem hard to solve.

To the best of our knowledge, only two works have investigated the optimal cache placement and sizing in the case of reactive content placement in arbitrary network topologies, although both focus on the specific case of on-path request routing with content placement in every node, which is known to generally yield poor caching performance. In [38], Rossini and Rossi investigate the performance of a heuristic approach which assigns cache capacity to nodes proportionally to certain centrality metrics. They concluded that these simple optimizations do not bring considerable performance advantages though. In [39], Wang *et al.* formulate the optimal cache allocation problem as a standard knapsack problem with constrained cumulative cache size. Despite their solution is not provably optimal, they reported that for the scenarios investigated, this cache allocation strategy improved upon the heuristics proposed in [38].

In the specific case of edge caching, since each request traverses a single cache, the problem is significantly simplified. In this situation, the optimal cache placement can be mapped to a p-median location-allocation problem [40].

Our proposed framework, similarly to edge caching, can be easily modeled since each request traverses only a single cache. As a result, our proposed approach has predictable performance and is also robust to variations in traffic patterns. In addition, as we show in section VI, the optimal cache placement problem can be solved optimally and in polynomial time for arbitrary topologies.

III. Overall framework design

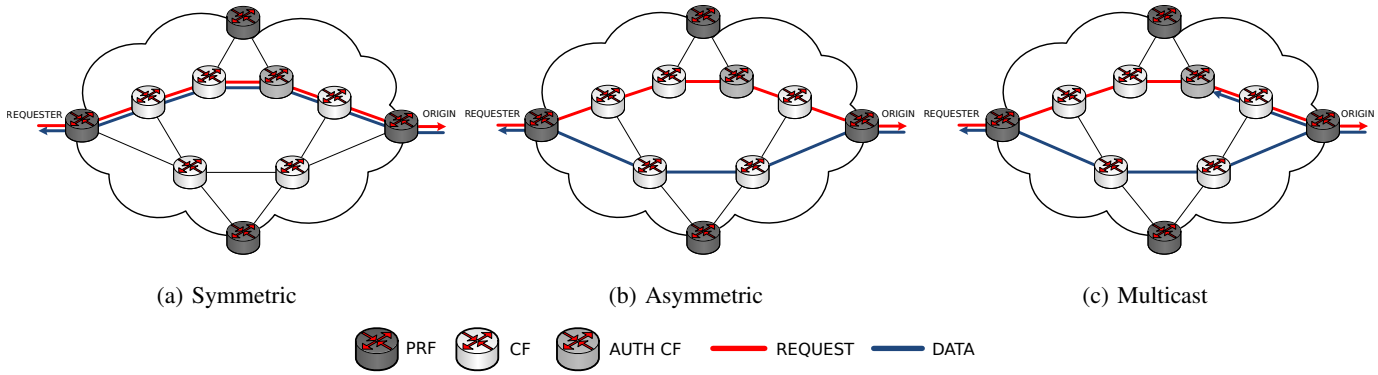
Our proposed framework comprises two types of functional entities: *Caching Functions (CF)* and *Proxy and Routing Functions (PRF)*. CF and PRF entities can be physically separated or co-located, but since, at least in principle, they provide separate functionalities we describe them separately.

- **Caching Functions (CF)** are simply in charge of storing the content items mapped to them by a shared hash function and serving incoming requests.
- **Proxy and Routing Functions (PRF)** are the first entities traversed by content requests when entering the cache network and they are in charge of forwarding them to relevant CF instances. They have knowledge of all CF instances in the network and their addresses (but do not need to know which content items are stored at each CF) and share a common hash function to map content identifiers to CF instances. Such a hash function does not need to be collision-resistant. Instead, it is desirable for its output to be produced incurring minimal processing overhead. Consistent hashing [10] may also be used in order to minimize the number of content items being remapped as a result of CF addition or removal.

The operation of the framework is illustrated in Fig. 1a and is named *symmetric* because content delivery follows the reverse path of request routing. Extensions of the framework for *asymmetric* and *multicast* operation are depicted in Fig. 1b and Fig. 1c respectively and are presented in Section IV-A. Any request is initially forwarded to the closest PRF. The PRF identifies the CF responsible for that content by computing the hash function on the content identifier and forwards the request to that CF. If the latter has the requested content, it returns it to the user. Otherwise, the request is forwarded towards the content origin through an egress PRF without looking up any other cache along the path. When the content item is returned by the content origin, the egress PRF forwards it to the authoritative cache that stores it and finally forwards it to the requesting user. We remark that the only form of coordination required by this system is that PRFs and CFs need to know how to reach all CFs deployed in the ISP, which are expected to be in the order of 10^2 or at most 10^3, but not need to keep any per content item state. This coordination can be achieved trivially with standard techniques.

This framework can be easily implemented in both the Network CDN and ICN cases using common techniques. In the Network CDN case, user requests are forwarded to the closest PRF as a result of a DNS lookup, as commonly done in current CDNs [41]. Both PRFs and CFs are simply reverse HTTP proxies with persistent TCP connections among them, which are used to relay HTTP requests and responses. Differently, in the ICN case, user requests are forwarded to the closest PRF through name-based routing, which can be achieved by inserting appropriate forwarding entries in the FIBs of access ICN routers. A PRF can then redirect a request to the authoritative CF by encapsulating the request packet with a header containing a name that identifies the authoritative cache. For example, if a request needs to be forwarded to CF A, the PRF can prepend a request header for name /_cache/_a.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) Symmetric (b) Asymmetric (c) Multicast

Fig. 1: Hash-routing content routing schemes. With *symmeteric routing* (a), content is routed to the requester via the authoritative cache. With *asymmeteric routing* (b), content is routed to the requester via the shortest path. With *multicast routing* (c), response is multicast to the authoritative cache and to the requester.

All ICN routers on the path are configured to route names with the /_cache/_a prefix to the location of CF A. When CF A receives the request packet, it removes the prepended header and processes the encapsulated packet. This is a standard approach, commonly used in ICN request routing to steer packets through instances of network functions [42]. Content packets are forwarded on the reverse path taken by request packets, and hence through the authoritative cache, using state information stored in intermediate routers.

The design of this hash-routing framework exhibits two interesting properties. First, since a content item can be cached only by the CF instance resulting from the hash calculation, a specific content item can be cached at most in one node of the network. This maximizes the number of distinct items cached within the ISP network, reducing redundancy, and therefore reducing also inter-domain traffic and load at origins. Second, localized spikes in traffic occurring in a network region are implicitly spread across links and caching nodes by the hash computation in a way similar in principle to Valiant Load Balancing (VLB) [43]. In Sec. VII-C we show that our scheme can indeed achieve very good load balancing under real operational conditions.

A possible deployment model would consist in co-locating of PRFs and CFs at (a subset of) the operator's PoPs. PRFs could also be deployed separately from CFs. For example, they may be deployed in the access network (*e.g.,* in an eNodeB in the case of an LTE network or in a DSLAM or BRAS in an xDSL fixed broadband access network) as part of an edge computing deployment. In this case, the PRFs deployed at the edge may also be equipped with a small caching space which is operated autonomously, *i.e.,* it can cache any content. We elaborate on this in Sec. IV-B.

We model this framework analytically in Sec. V. In the following section we present a set of content placement and request routing algorithms that can be applied to this framework to support additional desirable properties and provide knobs to fine-tune its performance. These extensions add complexity to the overall design and cannot, as such, be easily turned into a practical and tractable analytical model using known techniques from the literature. Therefore, their performance is only evaluated with trace-driven simulations.

IV. Content Placement and Request Routing

A. Asymmetric and multicast content routing

In the base design described above, request and content paths are symmetric, *i.e.,* in case of a cache miss, a content item (on its way back from the origin) is forwarded first to the cache and then to the requester, following the reverse path of the request. While this routing scheme is simple to implement and manage, the path stretch resulting from request and content routing through off-path caching nodes may lead to increased latency and link load. To address this limitation, we propose two alternative content routing schemes in addition to the base one proposed above, which we refer to as *Symmetric* (see Fig. 1a). These schemes, denoted as *Asymmetric* and *Multicast*, respectively, differ from the symmetric scheme only with respect to the delivery of contents entering the network as a result of a cache miss.

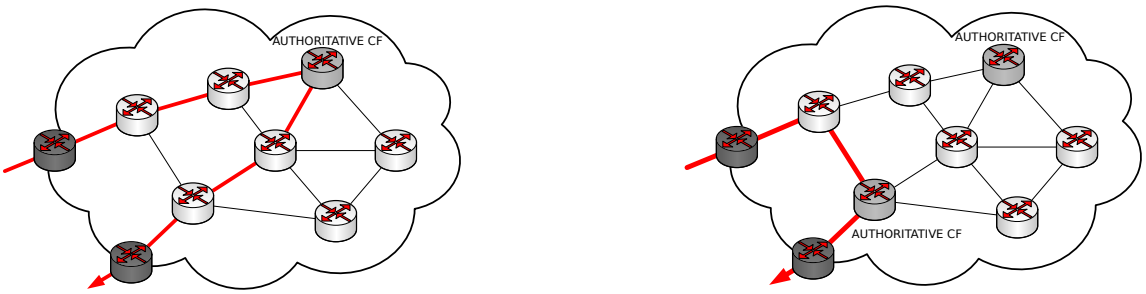
Asymmetric hash-routing (see Fig. 1b) routes contents always through the shortest path. The content is cached in the responsible cache only if it is on the shortest path between origin and requester, otherwise it is not cached at all.

Asymmetric routing reduces the overall network load since contents are always delivered though the shortest path. However, CF instances are populated with contents only if they are on the shortest path from origin to requester. This would not be a problem if content popularity varies slowly. On the contrary, it would actually increase cache hit ratio by reducing the impact of one-timers, as it would statistically require more requests for a content item to be cached. If, differently, request patterns exhibit strong temporal locality, a content item may need to be requested several times before it is actually cached (especially if the responsible cache is located on an underutilized path), hence affecting caching performance.

With *Multicast hash-routing* (see Fig. 1c), when the egress PRF receives a content, it multicasts (more accurately *twocasts*) it to both the authoritative cache and the requester. The target of multicast is to reduce delivery delay by sending one copy of the content through the shortest path back to the client, while leaving at the same time a copy of the content in the authoritative CF.

Multicast routing achieves the same latency of asymmetric routing and only requires one request for a content item to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) Single replica (b) Multiple replicas

Fig. 2: Single and multiple content replication. In large networks, replicating content only once may cause high path stretch to route requests and contents via the authoritative cache (a). Replicating content multiple times addresses the problem (b).

be cached, as in symmetric routing. However, since at each content request, a content item must be forwarded to the authoritative cache, it leads to greater link loads than the one achieved by asymmetric routing, especially when cache and requester paths have limited overlap.

Note that if the authoritative cache happens to be on the shortest path between content origin and requester, then symmetric, asymmetric and multicast schemes operate identically.

All three content routing options have advantages and disadvantages. The choice of content routing, however, gives network operators a knob to tune performance. Both multicast and asymmetric routing reduce latency because contents fetched from the origin are delivered to the requesting user without any detour. On the other hand, asymmetric routing caches the content in the domain only if the responsible cache is on the response path, while multicast, *i.e.,* twocast, routing increases complexity and traffic on the network. The choice of the most suitable scheme depends on the operator's priorities.

B. Hybrid caching

With *hybrid caching*, a part of the overall caching space is allocated to operate autonomously, *i.e.,* cache opportunistically any content, regardless of the content-cache mapping. In this case, a part of the CF caching space caches any content traversing it which is not part of the set of designated contents it has been assigned to cache.

This approach provides two main advantages. First, it allows a small number of very popular contents to be replicated in multiple nodes, instead of just one, hence possibly reducing overall latency and link load. Second, it achieves better load balancing across caching nodes. In fact, while the base hash-routing framework evens out localized traffic hotspots by spreading traffic originated from each requester across caches, any peak in demand for a specific content item will always be served by the same cache. In contrast, caching a small fraction of very popular contents in multiple caching nodes makes the system robust to variations in content popularity [44]. This also makes the system more robust to adversarial workloads targeted at overloading a specific caching node [45].

Additionally, in case asymmetric content routing is adopted, *hybrid caching* ensures that any requested content is cached at some node in the network (*i.e.,* not necessarily the authoritative CF) the first time it is requested. This is the case even if the responsible cache is not on the shortest path between origin and requester.

Performance can be tuned by selecting what fraction of caching space to dedicate to uncoordinated caching and this is a provisioning decision that should be made by the operator according to their priorities. Increasing the uncoordinated caching space leads to greater robustness towards flash-crowd events and lower latency in accessing the most popular contents. However, it reduces the amount of cache available to hash routing and as a result reduces the number of distinct items cacheable in the network. This affects overall cache hit ratio (possibly leading to a greater overall latency) and robustness against localized spikes in traffic.

C. Multiple content replicas

The base design of our framework allows a content item to be cached at most once in a network domain, hence maximizing the efficiency of caching space. However, in case of very large topologies, *e.g.,* of Tier-1 operators, this may result in high path stretch, possibly leading to high latency (see Fig. 2a). To address this issue, we propose a content placement and request routing algorithm where the content-to-cache hash function maps to k distinct nodes instead of just one. As a result, multiple nodes (ideally well distributed over the network) are responsible for each content (see Fig. 2b). This ensures that the worst case path stretch to reach a responsible cache is reduced.

When a PRF processes a content request and computes the content hash, it resolves the content to k distinct caching nodes. The PRF forwards the request to the closest of the k CFs responsible for the content. In case of cache miss, two approaches can be adopted:

1) The CF forwards the request directly to the content origin.
2) The CF forwards the request towards the content origin through (all or a subset of) the other CFs responsible for the content, if this can be done with limited path stretch.

Similarly, when a content enters the network after an origin fetch, several options can be adopted. The content can be forwarded symmetrically, asymmetrically or with multicast. In case of symmetric content routing, if several caches are looked up, contents can be placed using various on-path meta algorithms such as LCE, LCD [21] or ProbCache [23].

The degree of replication k enables the fine-tuning of performance by trading off latency with cache hit ratio. Hash-routing operating with multiple content replicas is a hybrid between hash-routing and on-path caching meta algorithms. At one extreme ($k = 1$) the system behaves as pure hash-routing scheme. At the other extreme (k equal to the number of caching nodes), the system behaves exactly as the on-path meta algorithms used for content routing. Sec. VII-E evaluates the impact of varying k on various performance metrics.

V. System modeling

One advantage of our framework over prior work is its performance predictability, that makes it considerably simpler to model and to optimize than other schemes. This section presents a thorough modeling of the hash-routing framework presented in Sec. III and derives closed-form expressions for content retrieval latency. We start by providing a generalized formulation for an arbitrary topology and then provide simpler expressions for two special cases: mesh and ring topologies. The modeling presented here will be subsequently used in the following section to design a polynomial algorithm for optimal cache node placement.

Consistently with previous work, we assume that content items are requested according to the Independent Reference Model (IRM). This implies that the probability that a specific item is requested is constant over time and does not depend on previous requests. We also assume, for simplicity, that all caches in the system have the same size, operate according to the same replacement policies and that content items are mapped to caching nodes uniformly.

We make two further assumptions based on findings from previous work. First, we assume that as a result of the uniform hashing, each caching node receives the same number of requests, independently of skewness in content popularity. Thaler and Ravishankar [9] show that this assumption holds as long as the content catalogue is large, which is the case of our interest. Second, we assume that the network-wide cache hit ratio is identical to the cache hit ratio yielded by a single cache with size equal to the cumulative size of all caches in network operating according to the same replacement policy. We showed in previous work [44] that this assumption holds for a variety of common replacement policies, including LRU, FIFO, RAND and LFU. The goodness of all these assumptions is demonstrated by the accuracy of the models presented below which heavily rely on them.

A. Arbitrary topology

We start by describing the notation adopted in this section, which is also used in the formulation of the optimal cache placement problem of Sec. VI and report it in Tab. I. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$ representing the operator's network and let $\mathcal{O}$ be the set of content items. A set of caches $\mathcal{C} \subseteq \mathcal{V}$ are deployed at a subset of network nodes. We assume that all caches have the same capacity and operate according to the same replacement policy. Content items are mapped uniformly to caches. Each node $v \in \mathcal{V}$ issues requests for content $o \in \mathcal{O}$ with rate $\lambda_o^{(v)}$. The cumulative rate of requests originated at node v is $\lambda^{(v)} = \sum_{o \in \mathcal{O}} \lambda_o^{(v)}$. The network-wide request rate is $\Lambda = \sum_{v \in \mathcal{V}} \lambda^{(v)}$. Each content item is permanently stored at one location outside the network and denote $\mathcal{O}_s$ as the set of content items stored at origin node s. We denote $\mu^{(s)}$ as the rate of miss requests forwarded towards the content origin via egress node s. We denote the system-wide cache hit ratio of content item $o \in \mathcal{O}$ as h_o, which can be estimated using known techniques (*e.g.,* Che's approximation [35], [36]) to model the network of caches as a single large cache [44]. We then denote as δ_e the average latency of link $e \in \mathcal{E}$ and denote as $\mathcal{P}_{s,t} \subseteq \mathcal{E}$ the network path from node s to node t, *i.e.,* the set of edges traversed from s to t. We further denote as $\delta_{s,t} = \sum_{e \in \mathcal{P}_{s,t}} \delta_e$ the average latency of path (s, t). We assume that the latency does not depend on the size of the message, *i.e.,* it is dominated by queueing delay and propagation delay as opposed to transmission delay.

TABLE I: Summary of notation

Symbol	Notation
$\mathcal{G}$, $\mathcal{V}$, $\mathcal{E}$	Network graph, set of vertices, set of edges
$\mathcal{C}$	Set of caches
$\mathcal{O}$, $\mathcal{O}_v$	Set of content items, stored at origin v
$\lambda^{(v)}$, $\lambda_o^{(v)}$	Request rate from requester v, for item o
Λ	Overall request rate
$\mu^{(v)}$	Rate of cache misses towards origin v
h, h_o	Overall cache hit ratio, for content item o
$\mathcal{P}_{s,t}$	Shortest path from s to t
$\delta_{s,t}$, δ_e	Latency over path $\mathcal{P}_{s,t}$, over edge e

After discussing the notation, we can now present the latency model. Since we assumed that each cache in the network has the same cache hit ratio and that load is equally spread across caching nodes, we can easily derive the cumulative rate of requests $\mu^{(s)}$ served by origin s as:

$$\mu^{(s)} = \sum_{v \in \mathcal{V}} \sum_{o \in \mathcal{O}_s} \lambda_o^{(v)} (1 - h_o) \tag{1}$$

We can now determine the average latency perceived by a user to retrieve a content item D. This value corresponds, in case of cache hit, to the round-trip time between the requester r and the responsible cache c, denoted as $D_{r,c}$. In the case of cache miss, latency is increased by the round-trip time between cache c and content origin s, denoted as $D_{c,s}$. Therefore:

$$D = D_{r,c} + D_{c,s} \tag{2}$$

Each latency component can be easily derived and equals to:

$$D_{r,c} = \frac{1}{\Lambda |\mathcal{C}|} \sum_{r \in \mathcal{V}} \lambda^{(r)} \sum_{c \in \mathcal{C}} (\delta_{r,c} + \delta_{c,r}) \tag{3}$$

$$D_{c,s} = \frac{1}{\Lambda |\mathcal{C}|} \sum_{s \in \mathcal{V}} \mu^{(s)} \sum_{c \in \mathcal{C}} (\delta_{c,s} + \delta_{s,c}) \tag{4}$$

While this model is relatively simple and provides a closed-form expression for content retrieval latency, there are some topologies of practical interest for which a more simplified formulation can be derived. The following sections provide simpler models for mesh and ring topologies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management

B. Mesh topology

We derive a simplified formulation of content retrieval latency for the case of a full mesh topology, which can be used to model highly connected network topologies, such as ISP core networks.

In our model, the network comprises N caching nodes, all receiving the same demand from a set of requesters attached via an access link. Content origins are attached to a subset of $M \leq N$ nodes. Each content item is stored at one origin node and each origin is assigned the same number of contents. We assume that all access, internal and external links have the same latency, denoted as δ_a, δ_i and δ_e respectively.

Theorem 1. *The mean content retrieval latency in a fully connected mesh topology of N nodes and $M \leq N$ egress nodes is:*

$$D = 2\left[\delta_a + \frac{N-1}{N}\delta_i + (1-h)\left(\frac{N-M}{N}\delta_i + \delta_e\right)\right] \quad (5)$$

where h is the overall cache hit ratio and δ_a, δ_i and δ_e are the latency of access, internal and external links respectively.

Proof. The overall content retrieval latency D comprises three components, which are the round trip time between (i) requester and authoritative cache $D_{r,c}$, (ii) authoritative cache and egress node $D_{c,e}$ and (iii) egress node and origin $D_{e,o}$. The two latter delay components are incurred only in case of a cache miss.

Since content items are assigned to caches uniformly, the probability that the ingress node is also the cache responsible for a given item is $1/N$, while the request needs to be forwarded to another cache, which is connected directly with a link to the ingress node, with probability $\frac{N-1}{N}$. Therefore, the average round trip time between requester and cache is:

$$\begin{aligned} D_{r,c} &= 2\left(\delta_a + 0 \cdot \frac{1}{N} + \delta_i \cdot \frac{N-1}{N}\right) \\ &= 2\left(\delta_a + \delta_i \frac{N-1}{N}\right) \end{aligned} \quad (6)$$

The probability that the egress node corresponds to the authoritative cache is $\frac{M}{N}$, while the request needs to be forwarded to another egress node with probability $\frac{N-M}{N}$. Then, the round trip time between the cache and the egress node is:

$$D_{c,e} = 2\left(0 \cdot \frac{M}{N} + \delta_i \cdot \frac{N-M}{N}\right) = 2\,\delta_i \frac{N-M}{N} \quad (7)$$

Finally, the round trip time between the egress node and the origin is, by definition:

$$D_{e,o} = 2\,\delta_e \quad (8)$$

The overall delay can then be computed as:

$$D = D_{r,c} + (1-h)\left(D_{c,e} + D_{e,o}\right) \quad (9)$$

Substituting Eq. 6, 7 and 8 into Eq. 9, we obtain Eq. 5 □

C. Ring topology

Another simple topology of practical interest is a ring. This can accurately model the case of a metro network.

We make the same assumptions of the mesh case, except that, in this case, we assume that the network has only one egress node. This is the common case in real applications of ring topologies.

Theorem 2. *The mean content retrieval latency in a ring topology of N nodes is:*

$$D = 2\left[\delta_a + \overline{H}\,\delta_i + (1-h)(\overline{H}\,\delta_i + \delta_e)\right] \quad (10)$$

where:

$$\overline{H} = \begin{cases} \frac{N^2-1}{4N}, & \text{if } N \text{ is odd} \\ \frac{N}{4}, & \text{if } N \text{ is even} \end{cases} \quad (11)$$

and h is the overall cache hit ratio and δ_a, δ_i and δ_e are the latency of access, internal and external links respectively.

Proof. The proof is similar to the one for the full mesh case. However, differently to that case, caches are not all interconnected with direct links.

Let $\overline{H}$ be the average path length (in terms of hops) between two randomly selected nodes of a ring, also including the case where origin and destination coincide. This can be trivially derived as:

$$\overline{H} = \frac{1}{N}\sum_{j=0}^{N-1} \min\left(j, N-j\right) = \frac{1}{N}\sum_{j=1}^{N-1} \min\left(j, N-j\right)$$

If N is odd, $\overline{H}$ can be further simplified as:

$$\overline{H} = \frac{2}{N}\sum_{j=1}^{\frac{N-1}{2}} j = \frac{2}{N} \cdot \frac{1}{2} \cdot \frac{N+1}{2} \cdot \frac{N-1}{2} = \frac{N^2-1}{4N}$$

while if N is even:

$$\overline{H} = \frac{1}{N}\left(\frac{N}{2} + 2\sum_{j=1}^{\frac{N}{2}-1} j\right) = \frac{1}{2} + \frac{2}{N}\left[\frac{N}{4}\left(\frac{N}{2} - 1\right)\right] = \frac{N}{4}$$

It can be easily observed that the mean round trip time between a requester and an authoritative cache $D_{r,c}$ and between authoritative cache and egress node $D_{c,e}$ are identical and are equal to:

$$D_{r,c} = D_{c,e} = 2\left(\delta_a + \overline{H}\,\delta_i\right) \quad (12)$$

while the round trip time between the egress node and the origin is identical to the case of a mesh network:

$$D_{e,o} = 2\,\delta_e \quad (13)$$

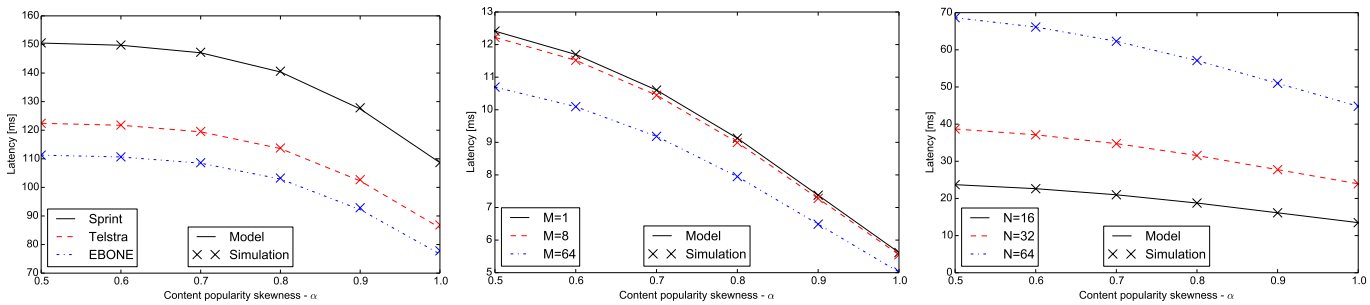
As in the mesh case, the overall delay can then be computed as:

$$D = D_{r,c} + (1-h)\left(D_{c,e} + D_{e,o}\right) \quad (14)$$

Substituting Eq. 12 and 13 into Eq. 14, we obtain Eq. 10.

□

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) Arbitrary topology (b) Mesh topology (c) Ring topology

Fig. 3: Accuracy of latency models

D. Model validation

To demonstrate the accuracy of the three models presented above, we show in Fig. 3 latencies measured from simulations and compare them with results from the models. In Fig. 3a we show the accuracy of the model for arbitrary networks using three real ISP topologies from the RocketFuel dataset [46]. In Fig. 3b we show the accuracy of the mesh model using a mesh of 64 nodes and a variable number of egress nodes M. Finally, in Fig. 3c we show the accuracy of the ring model using a ring topology with a variable number of nodes N.

In all scenarios, performance is evaluated using a stationary workload of 1M content items and Zipf-distributed content popularity with exponent α varying from 0.5 to 1. We provision networks with a cumulative cache space equal to 10% of the content population, *i.e.,* 100K items, equally distributed among caching nodes, all operating according to the LRU replacement policy. Simulations were executed issuing 5M requests to warm up caches followed by further 5M requests over which measurements were collected. We repeated each simulation 10 times using different random seeds and computed the 95% confidence interval using the repetitions method. We did not plot error bars because they were too small to be distinguishable from point markers.

All graphs show an excellent agreement between model and simulation results under all parameters considered. This proves not only the accuracy of the latency models, but also the goodness of our assumptions about cache hit ratio and load balancing upon which these models are based.

VI. Optimal cache placement

Based on the modeling presented above we designed a polynomial-time algorithm to compute the optimal placement of caching nodes across a network given the target number of caching locations p, the size of a single caching node S and the global demand. Its objective is the minimization of the overall latency assuming, consistently with the rest of the paper, homogeneous cache hit ratio among caches and uniform load balancing among caching nodes.

The algorithm, which we present using the same notation used in Sec. V-A, is defined in Fig. 4. The overall cache hit ratio h, which is an input of the problem, can be easily estimated either analytically using for example the characteristic time approximation [35], [36] or via simulation. As it can be easily observed from its formulation, the algorithm essentially consists in placing caches into the candidate nodes with lowest weighted latency towards requesters and origins. This simple formulation is a results of the hashrouting property that each request traverses a single cache, which makes modeling of cache dynamics considerably more tractable than algorithms where each request can traverse an arbitrary number of caches.

Input:

p number of caching nodes to deploy

h overall cache hit ratio of a cache of size pS subject to global demand

Output:

cache deployment configuration

Procedure:

1) Compute for each cache candidate node v the cost d_v defined as:

$$d_v = \frac{1}{\Lambda} \sum_{u \in \mathcal{V}} \lambda^{(u)} \delta_{u,v} + \mu^{(u)} \delta_{v,u}$$

2) Sort cache candidate nodes according to their cost d_v
3) Place caches on the p nodes with lowest cost d

Fig. 4: Optimal cache placement algorithm

We now prove that the solution provided by this algorithm is optimal and that the algorithm runs in polynomial time.

Proof of optimality. The cost d_v corresponds to the average latency experienced if node v was the only cache in the system with size pS. This can be proved immediately comparing its definition with Eq. 3, 4 and 2. As per the assumptions laid out above, the overall latency D of a network can be computed as:

$$D = \frac{1}{p} \sum_{v \in \mathcal{V}} d_v X_v$$

where $X_v \in \{0, 1\}$ takes value 1 if a cache is deployed in node v and 0 otherwise. It is obvious that the solution that minimizes D satisfying the constraint $\sum_{v \in \mathcal{V}} X_v = p$ consists in setting $X_v = 1$ for the p nodes with the lowest cost d_v and $X_v = 0$ for the remaining nodes. □

TABLE II: Network topologies

ASN	Name	Region	# PoPs	# backbone links
1221	Telstra	Australia	104	150
1239	Sprintlink	US	315	971
1755	EBONE	Europe	87	160
3257	Tiscali	Europe	161	327
3967	Exodus	US	79	146
6461	Abovenet	US	138	371

Proof of polynomial time equivalence. The execution of the algorithm requires in step 1 the calculation of $|\mathcal{V}|^2$ sums, in step 2 the sorting of $|\mathcal{V}|$ values that can be done in at most $O\left(|\mathcal{V}|\log|\mathcal{V}|\right)$ or $O\left(|\mathcal{V}|^2\right)$ steps depending on the algorithm used and finally in step 3 the assignment of $p < |\mathcal{V}|$ variables. It can be trivially noted that the overall number of steps to be executed is $O\left(|\mathcal{V}|^2\right)$. Therefore the algorithm has polynomial time complexity with respect to the number of nodes $|\mathcal{V}|$. □

The number of caching nodes to deploy, p, is an input of the optimization problem, which can be set according to the requirements of the operator. For example, we can use the smallest p that guarantees that each caching node is able to handle the expected peak load. In this case, if R is the overall peak rate of requests expected by the system and r is the maximum rate of requests that a single caching node can handle, then we can select $p = \lceil R/r \rceil$.

In our problem formulation we do not account for link capacity constraints. The rationale for this decision is that in normal operational conditions, intradomain network links operate at a small fraction of their capacity, as we learned from discussions with a large European ISP operating a CDN infrastructure. Links operate at high load conditions only in the case of transient traffic spikes, which are implicitly addressed by our framework. We validate the robustness of our solution in terms of link load balancing in Sec. VII-C.

VII. Performance evaluation

In this section we present the results of our performance evaluation. We investigate the performance of the algorithms described in this paper and compare them against previously proposed techniques under a variety of operational conditions. We also present how effectively our framework helps fine-tuning performance.

A. Setup and methodology

We evaluate the performance of our proposed framework with trace-driven simulations, using the *Icarus* simulator [47]. We make publicly available the code required to reproduce the results of this paper in [11].

We performed our analysis using various combinations of real traffic traces and topologies.

We used six PoP-level ISP topologies from the RocketFuel dataset [46] annotated with inferred link weights and latencies (see Tab. II). These six topologies provide a good variety of network sizes (from 79 to 315 PoPs) and refer to different geographic regions (US, Europe and Australia). We generate content requests from clients attached to each PoP with equal rate and assume that all content origins are outside the ISP. We set the cumulative cache size of the ISP to be equal to 0.1% of the content catalogue size. Each caching node is equipped with the same cache size and replaces content items according to the LRU policy.

TABLE III: Traces

Trace	# requests	# items	Zipf α	Duration
Wikipedia	11,566,029	1,834,747	0.99	1d
IRCache (all)	8,278,100	5,240,029	0.70	2d
IRCache (UIUC)	5,925,463	3,964,700	0.70	1w

We used workloads derived from three real traces (see Tab. III). For each of these traces, we used the first 25% of requests to warm up caches and measured performance over the remaining 75% of requests. The first workload we used is a 1-day trace containing a sample of 10% of all requests received by Wikipedia [48]. The other two workloads are from the IRCache dataset [49], which contains all HTTP requests collected by a set of proxies deployed at several universities in the United States. We used two traces from this dataset. The first, which we labeled *IRCache all*, is a 2-day trace collecting all requests coming from seven different sites. The second, labeled *IRCache UIUC* is a 7-day trace containing only requests coming from the proxy deployed at the University of Illinois Urbana-Champaign.

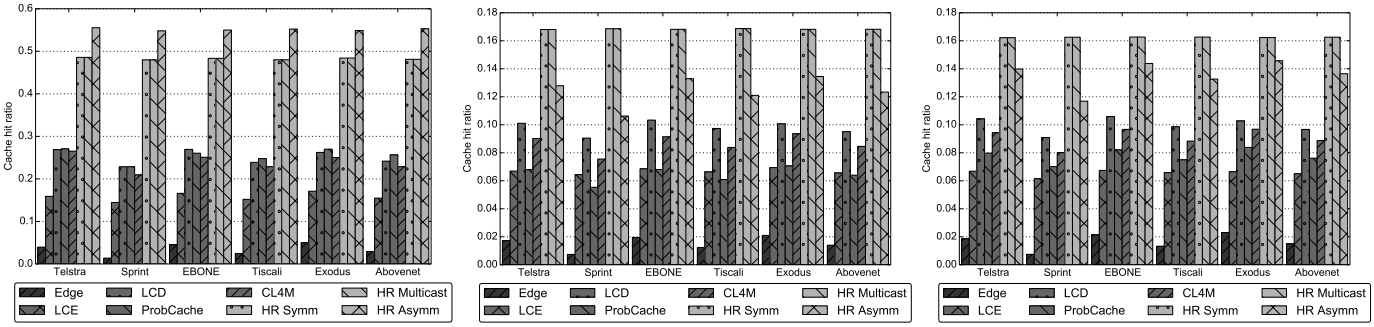
The traces used for our experiments have different characteristics. The Wikipedia trace has a more skewed content popularity distribution than IRCache ones, as evidenced by the greater value of Zipf α parameter (0.99 *vs.* 0.70). All traces have a similar amount of requests (6M, 8M and 11M) but over different time periods (1, 2 and 7 days). As a result, they have different temporal locality characteristics. Wikipedia is the most stationary while IRCache UIUC is the most dynamic. Finally, the two IRCache traces, despite having similar content popularity skeweness, cover a different geographic area. IRCache ALL encompasses requests from seven different regions, while the IRCache UIUC only from one single campus. We believe this mix of traces covers a large variety of realistic operational conditions.

We report that for each experiment requiring randomized configuration (*i.e.,* assignment of contents to origin nodes), we repeated each experiment 10 times and computed the 95% confidence interval using the repetitions method. We omitted to plot error bars if too small to be distinguishable from point markers. Finally, due to space limitations, we present results for all 18 topology-workload combinations at our disposal only in Sec. VII-B. In all other cases, we report that we observed similar results among all combinations. Therefore we only present results for the Wikipedia workload, which we selected because this dataset has the largest number of requests. Instead, the Telstra, Sprint and EBONE topologies were chosen because they cover three different geographic areas (Australia, US and Europe) and also have different sizes.

B. Base hash-routing

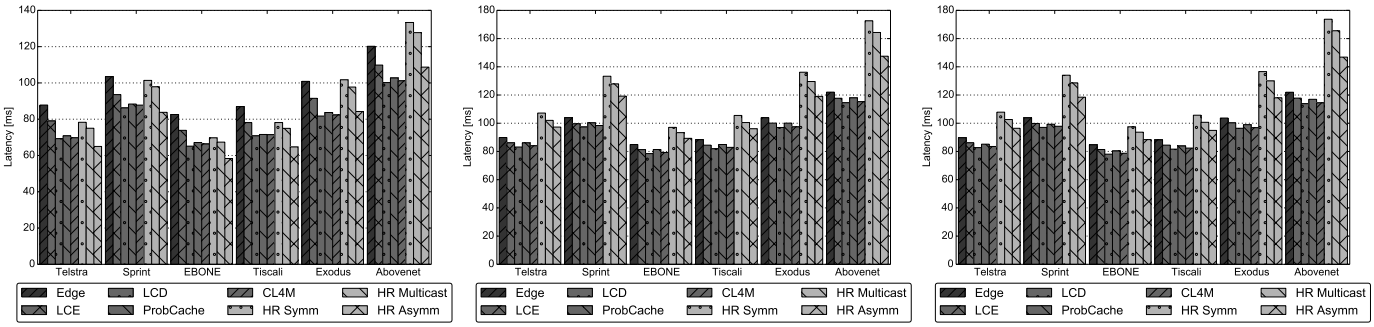
We begin by analyzing the performance of the three content routing schemes proposed (*i.e.,* symmetric, asymmetric and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) Wikipedia (b) IRCache (all) (c) IRCache (UIUC)

Fig. 5: Cache hit ratio



(a) Wikipedia (b) IRCache (all) (c) IRCache (UIUC)

Fig. 6: Latency

multicast) and compare them against other content placement strategies across various topologies and workloads. Since for most of the content placement techniques that we use as baseline there are no optimized cache placement algorithms, we present the results referring to a dense cache deployment (*i.e.,* a cache for each PoP). The results presented here are simply a lower bound of the performance that our framework can achieve, since, as we show later, optimal cache placement further enhance performance.

Fig. 5 shows that hash-routing schemes achieve a considerably higher cache hit ratio compared to other caching schemes, across all topologies and workloads considered. This is expected since hash-routing, differently from other schemes, maximizes the number of distinct content items cached in the network. Symmetric and multicast routing achieve identical cache hit ratio by design, since in both cases each cache miss results in the insertion of the requested content in the responsible cache.

On the other hand, asymmetric routing achieves better cache hit ratio than symmetric/multicast routing in the case of Wikipedia workload and worse in the case of IRCache workloads. This difference is explained by the fact that IRCache workloads have a lower density of requests and span a larger time period and hence exhibit a less stationary behavior than Wikipedia traces. With asymmetric routing, content items are inserted in cache only for a fraction of cache misses, as if each caching node operated with probabilistic insertion. This is known to reduce the impact of unpopular contents on LRU caches leading to a higher steady-state cache hit ratio but slower convergence. This justifies the better performance with more stationary workloads and worse performance with more dynamic ones.

With respect to latency (see Fig. 6), we notice that performance achieved by hash-routing schemes are pretty similar to those achieved by on-path schemes. Under all workloads and topologies considered, symmetric routing is slower to deliver content compared to multicast and asymmetric schemes that always forward content items over the shortest path to the requester.

The main takeaway is that, *even in an unoptimized cache placement scenario, hash-routing algorithms perform significantly better in terms of cache hit ratio (which also leads to reduced OPEX as a result of lower interdomain traffic) while still yielding latency values similar to other schemes.*

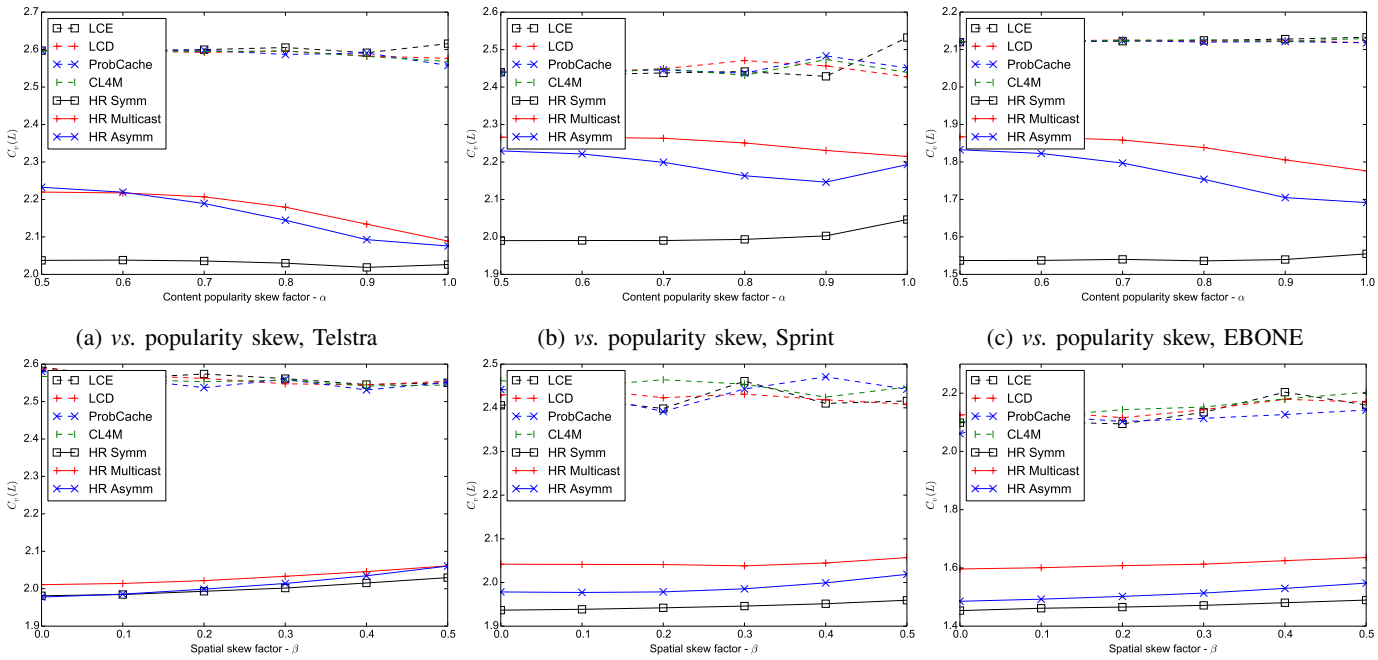
C. Load balancing

In this section, we evaluate the load balancing properties of our proposed framework, which is designed to balance load implicitly by uniformly spreading traffic over caching nodes similarly in principle to Valiant Load Balancing (VLB) [43].

We focus on load imbalance across network links. In line with previous work [9], [44], we quantify it using the coefficient of variation[1] of link load across all links of the network which we denote as $c_v(L)$. In addition to analyzing how well traffic is balanced across links, we also investigate robustness against variations in content popularity and geographic distribution of requests.

[1]The coefficient of variation of a random variable L is the ratio between its standard deviation and its mean value: $c_v(L) = \sqrt{\text{Var}(L)}/E[L]$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) *vs.* popularity skew, Telstra (b) *vs.* popularity skew, Sprint (c) *vs.* popularity skew, EBONE

(d) *vs.* spatial skew, Telstra (e) *vs.* spatial skew, Sprint (f) *vs.* spatial skew, EBONE

Fig. 7: Load balancing across links *vs.* popularity and spatial skew

In order to investigate load balancing against variations in content popularity, we run experiments in which each PoP generates requests with equal rates and content popularity following a Zipf distribution with coefficient α that we vary between 0.5 and 1.0. The greater the value of α, the more skewed the content popularity distribution. Instead, to investigate sensitivity against geographic heterogeneity in content request rates (and hence against the presence of request hotspots), we run experiments using the Wikipedia workload but with the rate of requests generated by each PoP not uniformly distributed, as in the case above. On the contrary, rates are assigned to PoPs following a Zipf distribution with coefficient β that we vary between 0 and 0.5. If $\beta = 0$, all PoPs originate requests with equal rates. Incrementing the value of β increases the difference between request rates across PoPs.

In Fig. 7, we present the results focusing on three RocketFuel topologies. From the graphs it can be immediately observed that all three hash-routing schemes achieve considerably better load balancing (*i.e.,* lower coefficient of variation) than all other caching schemes for all topologies and all values of α and β considered. Among hash-routing schemes, symmetric content routing yields the lowest load imbalance. This is justified by the fact that, differently from other schemes, in case of cache miss all contents are always delivered through the responsible cache, hence leading to a greater scattering of traffic across network links. Finally, it is also worth noting that hash-routing plot lines are mostly horizontal, meaning that load imbalance does not vary significantly as α and β change. This shows that *hash-routing load balancing, in addition to achieving better performance than previous proposals, is also robust against variations in both content popularity and geographic distribution of requests.*

D. Optimal cache placement and sparse deployment

In this section we show the latency improvement provided by an optimal placement of caches according to the algorithm presented in Sec. VI as opposed to a random placement.

To add a further element of comparison, we also analyze the latency achieved by edge caching with optimal cache placement. In this case the optimal placement can be mapped to a p-median location-allocation. We solve it using the Adjusted Vertex Substitution (AVS) algorithm [50], which is a recently proposed improvement of the well-known vertex substitution/interchange algorithm [51].

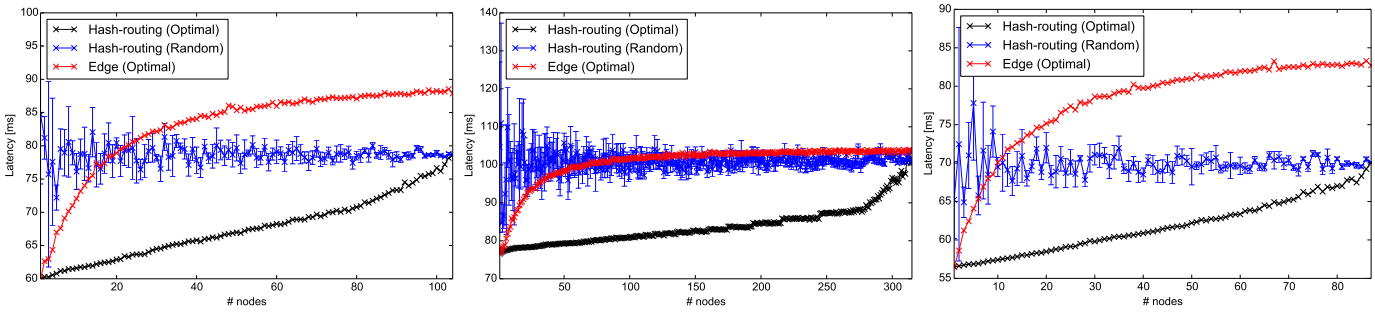
Results, depicted in Fig. 8, show first of all that *optimal cache placement strongly reduces latency for all topologies and number of caching nodes considered*. When caches are deployed over few nodes, the latency achieved by random placement strongly depends on the specific realization, as shown by the large error bars, which represent standard deviation across realizations. As the number of caching nodes increases, latency variability decreases, as the number of possible combinations diminishes and converges to the performance of optimal placement when every PoP has a cache.

In addition, these experiments show that optimized hash-routing achieves considerably lower latency than optimized edge caching for all topologies and number of nodes considered. This in particular demonstrates that *hash-routing is a good solution for both sparse and dense cache deployments*.

E. Multiple replicas

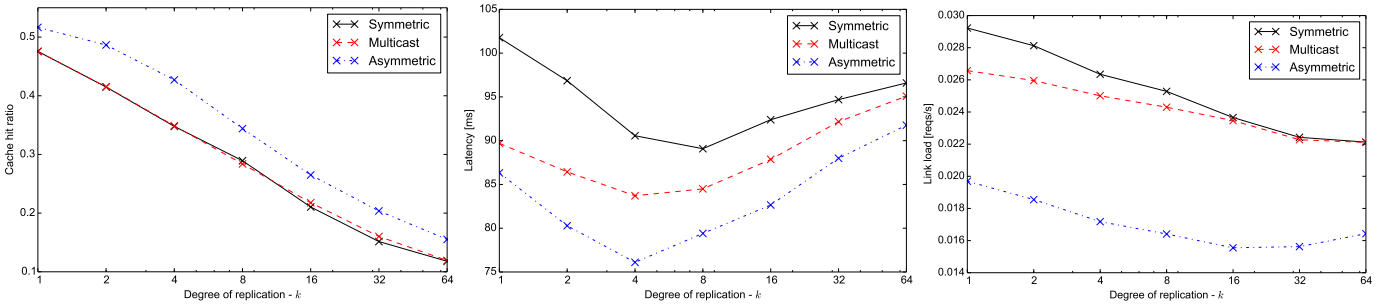
The path stretch incurred by hash-routing schemes for routing traffic through off-path caches may impact latency, especially in very large topologies. Although the results depicted in Fig. 6 show that even in large topologies hash-routing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2956525, IEEE Transactions on Network and Service Management



(a) Telstra (b) Sprint (c) EBONE

Fig. 8: Performance of cache placement algorithms, Wikipedia workload



(a) Cache hit ratio (b) Latency (c) Link load

Fig. 9: Performance of multiple replicas, Sprint topology and Wikipedia workload

latencies are comparable to those achieved by other caching schemes, we argue that it would be beneficial to have knobs allowing operators to further reduce latency, possibly trading off cache hit ratio. For this reason, we proposed in Sec. IV-C a mechanism enabling multiple caching nodes to be responsible for each content. In this section, we evaluate the performance of hash-routing schemes with multiple content replicas and show that by replicating content items more than once it is in fact possible to improve latency by trading cache hit ratio.

The problem of efficiently assigning content items to caches with multiple replicas is considerably more complex than the single replica case. For practical purposes, we propose a simple placement algorithm to allocate content replicas. First, we set the degree of replication k (*i.e.,* the number of caching nodes responsible for each content items). Then we group nodes in clusters in such a way that the latency among nodes of the same cluster is minimized. We do so by applying the Partitioning Around Medoids (PAM) clustering algorithm [52]. Then we map content items to each cluster independently, so that in each cluster there is one cache responsible for each content and, therefore, in the entire network there are k caches responsible for each content, one per cluster. It should be noted that this algorithm does not ensure that each cluster has the same or even a comparable number of nodes.

We experimented applying this placement to the largest topology of the RocketFuel dataset (*i.e.,* Sprint) and testing it with the Wikipedia workload. We investigated two routing options: (i) querying all responsible caches on the cluster-level shortest path from requester to origin and (ii) querying only the responsible cache belonging to the same cluster of the requester and, in case of a miss, routing the request to the origin without querying any other cache.

We observed (but do not present here due to space limitations) that the first option generally yields poor results as it provides only a marginal reduction in path stretch, not sufficient to compensate the loss in cache hits. As a result, increasing the number of clusters leads to both a reduction of cache hits and an increase of latency and link load. Nevertheless, we do not rule out the possibility that a more sophisticated content-to-cache assignment algorithm could provide better results. We reserve this analysis for future work.

On the other hand, the second option, *i.e.,* querying only the responsible cache, provides very interesting results, which we plot in Fig. 9. As shown in the figure, increasing the number of replicas reduces the cache hit ratio. This is expected since the number of distinct content items that the domain can cache declines. An interesting aspect is that symmetric and multicast content routing do not necessarily yield identical cache hit ratios if $k > 1$. This occurs because, in case of a cache miss, content items are inserted in the responsible cache of the cluster to which the requester belongs as well as responsible caches on other clusters if they occur to be on the delivery path. This opportunistic insertion in caches of other clusters depends on the content path, which differs between symmetric and multicast routing.

Regarding latency, we identify a local minimum corresponding to $k = 4$ for asymmetric and multicast and $k = 8$ for symmetric content routing. Further increasing the number of clusters leads to worse latency because decreasing cache hit ratio is no longer offset by a substantial reduction in path stretch.

Link load also decreases by increasing the value of k.

However, differently from the case of latency, there is a local minimum only for asymmetric routing (for $k = 16$), while for symmetric and multicast routing, link load always decreases as the number of clusters increases.

The main takeaway is that *multiple replication of contents is effective in providing a knob to trade off cache hit ratio with latency. A small degree of replication is sufficient to achieve a substantial latency reduction even using simple placement algorithms such as the k-medoid clustering we proposed.*

VIII. Related work

While we already discussed work related to content caching and cache placement in Sec. II, we provide here a brief overview of previous work specifically regarding hash-routing techniques to better understand the context of our contribution.

As already discussed, hash-routing is a well known technique widely used in Web caches albeit only in the context of co-located caches [8], [53]. In this context, a number of studies focused on designing hash functions that minimize the number of content items to be remapped as a result of servers being added or removed to a cluster. Notable proposals include rendezvous hashing [9] and consistent hashing [10]. These techniques achieve similar results but using different methods and can be equally applied to the case of geographically distributed caches.

Subsequent to our previous work [7], further proposals have investigated the application of hash-routing techniques to distributed environments, albeit limitedly to the context of ICN. Among those there is CoRC [54], which proposes to use hash-routing techniques among ICN routers of a domain like our framework, but with the objective of reducing the size of a forwarding table that each router needs to store by partitioning it across routers. Saha *et al.* [55] proposed the use of hash-routing for cooperative caching but focusing on the specific use case of inter-AS caching without taking intra-AS design considerations into account, such as load balancing and optimization aspects, which our proposal addresses. Therefore their approach is complementary to ours which focuses on intra-AS caching. Finally Wang *et al.* [56] formulated an optimization algorithm for the assignment of multiple content replicas to caching nodes operating under symmetric hash-routing. However, their work requires all caches to operate according to the LFU replacement policy, which is in practice scarcely utilized due to its poor reactivity to changes in content popularity. Differently, the modeling and optimal cache placement of our framework is applicable to a wider range of commonly used cache replacement policies, including LRU, FIFO, RANDOM and also LFU.

IX. Summary and conclusions

We presented a framework for optimal cache and hash-routing techniques for use in geographically distributed networks of caches. We showed that content hash-routing schemes provide several advantages in comparison to state-of-the-art techniques.

We showed that the proposed solution achieves excellent cache hit ratio, hence reducing transit costs and also contributes to reduced content retrieval latency. However, most importantly and differently from other schemes, hash-routing techniques have a set of key advantages. First, they are extremely robust against rapid variation of traffic patterns and traffic spikes, both in terms of specific cache locations and in terms of popular content. Second, they distribute traffic evenly across a network domain, *de facto* eliminating the possibility of hot spots. Third, they are easier to model and as a result provide predictable performance and make cache placement easier. Their performance predictability and robustness makes it easier for operators to meet SLAs agreed with customers. Finally, they provide several knobs to fine-tune performance metrics depending on the target topology, deployment models and expected workloads.

Our proposed content placement and routing framework allows the operator to decide among different variations of content caching and routing within its domain (symmetric, asymetric, multicast), potentially assign a part of the caching space for opportunistic caching in a hybrid fashion together with the targetted content caching proposed by our framework, and in the case of a large topology, e.g. for Tier-1 ISPs, potentially support multiple content replicas to reduce latency. We believe this to be a holistic framework applicable to both emerging operator-provided CDNs and also to future ICNs if these see deployment in the longer term future.

Acknowledgment

This work was supported by the European Commission Horizon 2020 ICN2020 project (GA no. 723014) and the UK Engineering and Physical Sciences Research Council (EPSRC) INSP fellowship (EP/M003787/1).

References

[1] "Cisco Visual Networking Index (VNI)," https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.
[2] "Verizon Digital Media Services," https://www.verizondigitalmedia.com/.
[3] "AT&T CDN Services," http://goo.gl/KxmVAm.
[4] "Level3 Content Delivery Network," http://goo.gl/p9AyhC.
[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'09)*. New York, NY, USA: ACM, 2009, pp. 1–12.
[6] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
[7] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *3rd ACM SIGCOMM workshop on Information-Centric Networking (ICN'13)*, Hong Kong, China, Aug. 2013.
[8] K. Ross, "Hash routing for collections of shared Web caches," *IEEE Network*, vol. 11, no. 6, pp. 37–44, Nov 1997.
[9] D. G. Thaler and C. V. Ravishankar, "Using name-based mappings to increase hit rates," *IEEE/ACM Transactions on Networking*, vol. 6, no. 1, pp. 1–14, Feb. 1998.
[10] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'97)*. New York, NY, USA: ACM, 1997, pp. 654–663.
[11] "Icarus simulator source code," https://github.com/icarus-sim/icarus.
[12] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. C. Schmidt, and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges," RFC 7927, Jul. 2016.

[13] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in Content Distribution Networks," *Comput. Oper. Res.*, vol. 35, no. 12, pp. 3860–3884, Dec. 2008.

[14] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, Aug. 2008.

[15] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proceedings of the 6th International COnference on Emerging Network Experiments and Technologies (CoNEXT'10)*. New York, NY, USA: ACM, 2010, pp. 4:1–4:12.

[16] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies ISP traffic engineering," in *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'13)*. New York, NY, USA: ACM, 2013, pp. 229–242.

[17] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM'10)*, March 2010, pp. 1–9.

[18] Y. Li, H. Xie, Y. Wen, C. Chow, and Z. Zhang, "How much to coordinate? optimizing in-network caching in content-centric networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 420–434, Sept 2015.

[19] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 286–299, September 2013.

[20] S. Imai, K. Leibnitz, and M. Murata, "Statistical approximation of efficient caching mechanisms for one-timers," *IEEE Transactions on Network and Service Management*, vol. 12, no. 4, pp. 595–604, Dec 2015.

[21] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Perform. Eval.*, vol. 63, no. 7, Jul. 2006.

[22] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758 – 770, 2013.

[23] I. Psaras, W. Koong Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2920–2931, Nov 2014.

[24] D. Tuncer, V. Sourlas, M. Charalambides, M. Claeys, J. Famaey, G. Pavlou, and F. D. Turck, "Scalable cache management for ISP-operated content delivery services," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2063–2076, Aug 2016.

[25] D. Fullagar, "Designing Netflix's Content Delivery System," 2014, uptime Institute Symposium. [Online]. Available: https://youtu.be/LkLLpYdDINA

[26] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 52–66, Jul. 2015.

[27] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of Facebook photo caching," in *Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP'13)*. New York, NY, USA: ACM, 2013, pp. 167–181.

[28] "Google Global Cache," https://peering.google.com/about/ggc.html.

[29] E. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM'09)*, April 2009, pp. 2631–2635.

[30] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," in *Proceedings of the 1st International Conference on Information-centric Networking (ICN'14)*. New York, NY, USA: ACM, 2014, pp. 127–136.

[31] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "Joint object placement and node dimensioning for Internet content distribution," *Inf. Process. Lett.*, vol. 89, no. 6, pp. 273–279, Mar. 2004.

[32] ——, "On the optimization of storage capacity allocation for content distribution," *Computer Networks*, vol. 47, no. 3, pp. 409 – 428, 2005.

[33] N. Belfari Melazzi, G. Bianchi, A. Caponi, and A. Detti, "A general, tractable and accurate model for a cascade of LRU caches," *IEEE Communications Letters*, vol. 18, no. 5, pp. 877–880, May 2014.

[34] P. R. Jelenković and X. Kang, "Characterizing the miss sequence of the LRU cache," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 119–121, Aug. 2008.

[35] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas of Communications*, vol. 20, no. 7, pp. 1305–1314, Sep. 2006.

[36] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFOCOM'14)*, April 2014, pp. 2040–2048.

[37] E. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM'10)*, March 2010, pp. 1–9.

[38] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proceedings of the 1st IEEE Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN'12)*, March 2012, pp. 280–285.

[39] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Transactions on Computers*, vol. PP, no. 99, 2015.

[40] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[41] F. Chen, R. K. Sitaraman, and M. Torres, "End-user mapping: Next generation request routing for content delivery," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*. New York, NY, USA: ACM, 2015, pp. 167–181.

[42] M. Arumaithurai, J. Chen, E. Monticelli, X. Fu, and K. K. Ramakrishnan, "Exploiting ICN for flexible management of Software-defined Networks," in *Proceedings of the 1st International Conference on Information-Centric Networking (ICN'14)*. New York, NY, USA: ACM, 2014, pp. 107–116.

[43] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC'81)*. New York, NY, USA: ACM, 1981, pp. 263–277.

[44] L. Saino, I. Psaras, and G. Pavlou, "Understanding sharded caching systems," in *Proceedings of the 35th IEEE International Conference on Computer Communications (INFOCOM'16)*. IEEE, 2016.

[45] B. Fan, H. Lim, D. G. Andersen, and M. Kaminsky, "Small cache, big effect: Provable load balancing for randomly partitioned cluster services," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC'11)*. New York, NY, USA: ACM, 2011, pp. 23:1–23:12.

[46] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02)*. New York, NY, USA: ACM, 2002, pp. 133–145.

[47] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (ICN)," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '14. ICST, Brussels, Belgium, Belgium: ICST, 2014.

[48] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Computer Networks*, vol. 53, no. 11, pp. 1830–1845, Jul. 2009.

[49] "The IRCache project," http://ircache.net/.

[50] J. Ashayeri, R. Heuts, and B. Tammel, "A modified simple heuristic for the p-median problem, with facilities design applications," *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 4–5, pp. 451 – 464, 2005.

[51] M. B. Teitz and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," *Operations Research*, vol. 16, no. 5, pp. pp. 955–961, 1968.

[52] L. Kaufman and P. J. Rousseeuw, *Partitioning Around Medoids (Program PAM)*. John Wiley & Sons, 2008, pp. 68–125.

[53] V. Valloppillil and K. W. Ross, "Cache array routing protocol v1.0," *Internet Draft draft-vinod-carp-v1-03.txt*, February 1998.

[54] H.-g. Choi, J. Yoo, T. Chung, N. Choi, T. Kwon, and Y. Choi, "CoRC: Coordinated Routing and Caching for Named Data Networking," in *Proceedings of the 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'14)*. New York, NY, USA: ACM, 2014, pp. 161–172.

[55] S. Saha, A. Lukyanenko, and A. Yla-Jaaski, "Efficient cache availability management in information-centric networks," *Computer Networks*, vol. 84, pp. 32 – 45, 2015.

[56] S. Wang, J. Bi, J. Wu, and A. Vasilakos, "CPHR: In-network caching for information-centric networking with partitioning and hash-routing," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.