Understanding Optimal Caching and Opportunistic Caching at "The Edge" of Information-Centric Networks

Ali Dabirmoghaddam[1] Maziar Mirzazad-Barijough[1] J. J. Garcia-Luna-Aceves[1,2]
[1]Computer Engineering Department, University of California, Santa Cruz, CA 95064
[2]PARC, Palo Alto, CA 94304
{alid,maziar,jj}@soe.ucsc.edu

ABSTRACT

A formal framework is presented for the characterization of cache allocation models in Information-Centric Networks (ICN). The framework is used to compare the performance of optimal caching everywhere in an ICN with opportunistic caching of content only near its consumers. This comparison is made using the independent reference model adopted in all prior studies, as well as a new model that captures non-stationary reference locality in space and time. The results obtained analytically and from simulations show that optimal caching throughout an ICN and opportunistic caching at the edge routers of an ICN perform comparably the same. In addition, caching content opportunistically only near its consumers is shown to outperform the traditional on-path caching approach assumed in most ICN architectures in an unstructured network with arbitrary topology represented as a random geometric graph.

Categories and Subject Descriptors

C.2 [**Computer Communication Networks**]: Network Architecture and Design; H.3 [**Information Storage and Retrieval**]: Systems and Software—*Information networks*

General Terms

Design,Theory

Keywords

information-centric networks; cache networks; network optimization; spatiotemporal locality of reference

1. INTRODUCTION

Several Information-Centric Networking (ICN) architectures [2,21] have been developed in an attempt to address the shift in the Internet communication paradigm from the conventional host-centric model towards a more flexible data-oriented design. As a result, ICN architectures seek to provide the necessary foundations for scalable and cost-efficient content distribution. A key design principle of many such architectures is the *universal in-network caching* of named data objects opportunistically. The *universality* of such opportunistic caching implies that it should be done *everywhere* and for *everything* in the network. The former requires all ICN routers to equally contribute in the network-wide process of caching, while the latter necessitates the ICN routers to cache all kinds of traffic they handle, irrespective of the popularity of the content or its geographical relevance. This approach is used to attain such performance benefits as reduced response time, efficient content distribution, and improved disruption tolerance.

As the review of prior work in Section 2 points out, even though universal in-network caching is assumed in many ICN architectures, there has been no quantitative analysis justifying this choice compared to opportunistic caching of content near its consumers. The main contribution of this paper is to provide a formal framework for the characterization of the performance of optimal in-network caching in ICNs, as well as opportunistic in-network caching at the edge of ICNs—*i.e.*, close to the end-users.

Section 3 uses the *Independent Reference Model* (IRM), which assumes that object references occur independently, to study the benefits of using universal caching compared to a simple policy of caching only at the edge of the network assuming a simple hierarchical caching structure. Our results, supported by extensive event-driven simulations over a wide range of configurations, indicate that the optimal caching approach based on universal caching provides only marginal benefits over the simple policy of caching only at the edge routers of the ICN. Although empirical studies [13] in the past have shown similar results, we present the first mathematical framework explaining this finding.

Section 4 addresses the impact of *locality of references* (*i.e.*, content requests) on the performance of caching in an ICN. Our work is inspired in part by the results delineated by Traverso *et al.* [31] on temporal locality of content references. We introduce a novel view of reference locality that captures both spatial and temporal aspects. The reference locality refers to the fact that a request to an object is likely to trigger subsequent requests from the same geographical neighborhood (*i.e.*, spatial locality) in the near future (*i.e.*, temporal locality). In other words, object references are localized in both space and time. Most prior work (*e.g.*, [4, 15, 26, 28]) neglects the existence of such dependencies by assuming the IRM model.

ICN'14, September 24–26 2014, Paris, France

http://dx.doi.org/10.1145/2660129.2660143

Exploiting the notion of cluster point processes [9], we present a general method to synthesize traces of object references while maintaining their locality properties. The procedure we use for generating such non-stationary traces complies with the intuitive perception of spread of epidemics on today's social networks. An information object first attracts attention in a specific geographical region. People start sharing the content with their social contacts. A subpopulation of their contacts who find the content interesting re-share it and this process is repeated so long as the content retains its informational value in the network. We leverage the fact that this process closely matches that of a self-exciting Hawkes process [19] and present a new algorithm to produce a synthetic trace in which, while the collective popularity profile of objects follows the commonly observed Zipf distribution [3,18], the occurrences of object-specific references over time and space are locally clustered when observed on a smaller scale. Based on this, we introduce a convenient measure to quantify the clustering degree of references on a scale from 0 to 1. We call this measure the *localization factor*, which can be used to cover the entire spectrum of reference patterns, from IRM (when it equals 0) to highly localized (when it goes to 1).

Armed with these new tools, we extend the comparison of universal in-network caching with simple caching at the edge of an ICN for traces not necessarily conforming to the IRM assumption. The results from our model in conjunction with event-driven simulations show that, while the optimal caching naturally drifts towards the edge as the caching budget increases, higher degrees of reference locality can further accelerate this transition. According to our findings, a 35% difference between edge- vs. optimal caching under the IRM assumption decreases to only 8% with a locality factor of 0.9.

Section 5 addresses the problem of caching in an unstructured ICN modeled using a random geometric graph. Given that optimal universal in-network caching is not possible to attain in this case, ICN architectures have adopted caching of content along the paths taken by content objects from producers or caches to consumers, which has recently been called Transparent En-Route Caching (TERC) [21]. The results from our simulations using ndnSIM [1] demonstrate that opportunistic caching at the edge of an ICN outperforms TERC in virtually all circumstances. While this result may be surprising at first, it can be explained with the insight gained by our modeling. TERC forces routers to store excessive amounts of content that induces much more content replacement along paths, while edge-caching tends to store more what is of interest to consumers near the routers.

Our work does not advocate specific mechanisms or ICN architectures. However, it provides new tools (e.g., the generation of meaningful synthetic traces) to analyze novel caching approaches in the future, and insight that has been missing to date on the caching schemes adopted in ICN architectures. In particular, given that universal in-network caching is not needed to attain efficiency, and given that edge-caching performs so well, new approaches should be developed that better integrate content routing and congestion control with content caching near consumers. Architecturally, our results indicate that deploying different types of routers in ICNs—some without any caching—would be far more cost effective. In the words of Fayazbakhsh *et al.* [13], content caching "at the edge" of ICNs indeed renders "less pain, most of the gain."

2. RELATED WORK

2.1 Caching

Although isolated caches have been studied extensively in the past (*e.g.*, [10, 20]), many aspects of interconnected networks of caches are not yet fully understood. Cache networks first became a subject of interest as a means to improve the performance of the Web [7, 27], and work on ICN architectures has renewed interest in this topic [4, 6, 16, 22–26, 33].

Understanding the full dimensions of networks of caches is naturally much more complicated than that of individual caches when operating in isolation. Many existing methods developed for analyzing the performance of isolated caches are based on algorithms that themselves are computationally expensive. For simplicity, these methods often introduce certain approximations that come at the inevitable cost of inaccuracy. Despite being negligible in the analysis of individual caches, these errors can aggregate and propagate through the system and produce a cascading effect when used in analysis of a tandem of caches.

A highly accurate approximation of least recently used (LRU) caching was introduced by Che *et al.* [7]. This analysis was recently revived in a seminal work by Fricker *et al.* [15] and shown to be applicable to a much wider range of scenarios beyond the specific conditions that Che *et al.* had initially anticipated. In the following, we briefly review this method which we shall refer to as "CHE-APRX"—abbreviated form of Che-approximation—hereinafter.

Consider a system comprising a total of N information objects and a LRU cache with capacity C. The requests for an object n come at the cache forming a Poisson process with rate $q(n)$. In fact, $q(n)$ signifies the popularity of object n in the system—*i.e.*, the proportion of total requests that belongs to n. The more popular an object n, the higher $q(n)$ as compared with other objects.

Che *et al.* define the *characteristic time* of a cache of size C, denoted by t_C, as the time it takes the cache to be filled with unique objects subject to the request rates $q(\cdot)$ under the IRM assumption, and show that t_C is indeed the unique root that solves the following equation for t:

$$C = \sum_{i=1}^{N} \left(1 - e^{-q(i)\,t}\right). \tag{1}$$

Knowing t_C, the miss probability $m(n)$ for an object n, according to CHE-APRX, is derived as:

$$m(n) \approx e^{-q(n)\,t_C}. \tag{2}$$

As mentioned earlier, the CHE-APRX has been proven to be very accurate and highly versatile. However, there are two important restrictions in this approximation.

1) Equal-sized objects. All information objects in CHE-APRX are of equal size—more precisely, unit-size such that the cache is able to store at most C objects. This assumption might seem far from reality at first, though becomes more plausible if objects are assumed to be segmented into equal-sized chunks, as required by many existing ICN proposals. It has also been shown [15] that CHE-APRX can readily be extended to also account for variable-sized objects. This, however, makes the derivations more unwieldy with little, if any, extra benefit to the purpose of our analysis. Hence, we choose to keep this assumption in place.

2) Independent object references. CHE-APRX assumes that the requests for information objects—a.k.a. references—arrive at the cache according to an i.i.d. process, independent of the past history of the requests and following a distribution determined by $q(\cdot)$ function. This assumption—generally referred to as the *Independent Reference Model* (IRM)—is fairly standard to many similar analyses for tractability and in order to calculate stationary hit/miss rates.

Although the IRM assumption is convenient, it is too simplistic in the context of cache networks, where object references exhibit strong correlation in both space and time domains. Consider for example a new song, while listed among the top hits of the month, may be highly popular for a certain period, but gradually gets faded out as newer hits are released. Furthermore, if the song is in a specific language, it may be well-received in certain regions of the world where that language is widely spoken, while attracting little attention in many other regions. The first example reflects the *temporal locality* of references, in contrast to the *spatial locality* highlighted by the second example. The IRM assumption disregards such localities in space and time by assuming that content popularity is stationary.

2.2 Architectures and Systems

In-network caching of named content is a cornerstone of many ICN architectures [2, 17, 21]. This consideration is so pervasive that many research papers (*e.g.*, [16, 21, 23, 29, 33]) use the notion of "cache network" as an abstraction to describe "content-centric networks".

The caching approach used in the vast majority of existing ICN proposals is the Transparent En-Route Caching (TERC) [21] by which *all* ICN routers in the network participate in the process of content caching in conjunction with their primitive function of relaying the information objects downstream. This naïve method of caching, however, has been subject of many controversies and criticisms [6, 13, 33]. To reduce caching redundancy, more complex varieties of this paradigm, such as probabilistic in-network caching (ProbCache) [24] and opportunistic caching using reinforced counters [12] have recently been introduced.

A handful of attempts in the past few years have been made to investigate most efficient methods of caching in ICN, both empirically [13, 31, 32] and analytically [4, 6, 24]. The results from some of these studies, however, are somewhat inconsistent and indecisive. For instance, the authors of [11] argue that caching at the *core* of the network can be more effective, as opposed to [13, 25] who advocate caching closer to the network *edge*.

Amidst this flurry of research, some researchers [6, 33] believe that the best cache placement strategy is greatly influenced by factors such as network topology; hence, there does not exist a unified strategy to be generally adopted. In contrast, other work [30] reports that the impact of topology on the performance of caching is limited.

Many notable analytical works [4, 15, 16, 26, 28, 33] focusing on the characterization of caching generally suffer from the limitations imposed by the IRM assumption. This assumption is so tightly coupled with the existing models of caching that Kurose writes [21]: "[The IRM] assumption is as fundamental for cache modeling as the memoryless assumption of exponential packet/circuit interarrival times ... are for modeling packet- and circuit-switched networks ..." This is indeed the case; however, just as exponential interarrival times, the IRM assumption is only a simplifying assumption to make problems tractable; there is no evidence showing that real-world traffic adheres to the IRM model [8, 14].

Recently, Traverso *et al.* [31] have addressed the importance of temporal locality of references in the performance of today's caching networks. Their work leverages the concept of Poisson shot noise processes as a convenient mathematical tool to model and analyze temporal reference locality. They further show that adopting the IRM assumption results in an overly pessimistic view of caching performance.

3. HIERARCHICAL CACHING MODEL

Given a hierarchical network of caches, we ask the question: How much should each layer of the hierarchy of caches contribute in the caching process in order to get the most out of a constrained total caching budget? We frame this question as an optimization problem and present solutions based on non-linear integer programming.

Consider a hierarchy of LRU caches in the form of a tree with its root acting as the content source. We assume that the source stores permanent copies of all the information objects in the system. Alternatively, the source can be considered as a collection of all possible content hosts that are logically collapsed into one single entity as the root of the tree in our model.

The tree comprises $L+2$ levels. The content subscribers (*i.e.*, users or information requesters) are at the 0^{th} level, while the content source is at level $L+1$. Subsequently, there exist L levels of nodes with caching capabilities between users and the content source which are sequentially labeled from bottom (level 1) to the top (level L).

The caching paradigm we seek to optimize is called "on-path caching" which works as follows. When a request for an object is raised at level 0, it is forwarded along the (unique) path of intermediate caches towards the root until a cache hit occurs. If all cache accesses are missed along the path, the request will be fulfilled by fetching a copy of the object directly from the source (root). Once located, the object is transferred on the reverse path back to the requester and a local copy is also stored on each and every cache along.

For simplicity, in the following analysis we assume that the hierarchy is a complete k-ary tree. Under the IRM assumption and given that the cache states are independent[1], the rate at which requests for an object n arrive at a particular cache at level ℓ can recursively be formalized as:

$$q_\ell(n) = \begin{cases} q(n) & \ell = 0\,, \\ k\, q_{\ell-1}(n)\, m_{\ell-1}(n) & 0 < \ell \leq L\,, \end{cases} \tag{3}$$

where $m_{\ell-1}(n)$ is the miss probability of object n at a cache of level $\ell-1$, which can itself be calculated directly using CHE-APRX (*i.e.*, Equations (1) and (2)).

3.1 The Expected Time To Access Content

A parameter of interest is the *expected time to access* (ETTA) an object n, which we denote by $\tau(n)$. This is defined as the expected duration between the time a user sends a request for an object n until a copy is located in the system (either at an intermediate cache or finally at the original source). We measure this duration in terms of the number of hops between the user and the closest replica of the content

[1]This assumption is reasonable when k is not very small.

along the path to the source. The following theorem gives a closed-form for calculating ETTA in terms of the miss rates of the intermediate caches.

Theorem 1. *Consider a tree structure with $L+2$ levels where the users are at level 0 and the content source at level $L+1$. Employing an on-path caching strategy as described before, the expected time to access an object n is obtained as:*

$$\tau(n) = 1 + \sum_{i=1}^{L} \prod_{j=1}^{i} m_j(n)\,, \tag{4}$$

where $m_j(n)$ is the miss probability of content n at a cache of level j on the path from the user towards the root of the tree.

Proof. For an object n, the problem can be modeled by a discrete-time Markov chain whose states are the levels of the caching hierarchy plus an additional state of H denoting a cache hit (See Figure 1). Every state ℓ, $0 < \ell < L+1$ transits into either the following state $\ell+1$ or H with probabilities $m_\ell(n)$ and $h_\ell(n) = 1 - m_\ell(n)$, respectively. States $L+1$ and H transit into state H with probability 1.

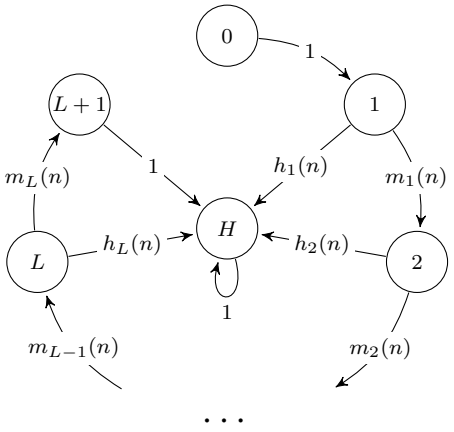

Figure 1: The Markov chain representing the process of locating an object on a cache tree. State 0 is where the users' requests are generated. State H denotes the state of a cache hit and other states correspond to the levels in the hierarchy from bottom to the top. State $L+1$ is the root of the tree where the content source is located. $m_i(n)$ and $h_i(n) = 1 - m_i(n)$ are the miss and hit probabilities at a cache of level i.

Define $T_H \triangleq \inf\{t \geq 1 : X_t = H\}$ as the stopping time denoting when state H is visited for the first time. Also, the expected time to visit H as $\tau_0(n) \triangleq \mathbb{E}[T_H \,|\, X_0 = 0]$, where X_0 is a random variable denoting the initial state. We note that $\tau_0(n)$ counts the expected number of transitions to visit state H which can be expressed recursively as:

$$\begin{aligned}\tau_0(n) &= 1 + \tau_1(n) \\ &= 1 + 1 + m_1(n)\,\tau_2(n) + \big(1 - m_1(n)\big)\,\tau_H(n)\,.\end{aligned}$$

Also, it is clear that $\tau_H(n) = 0$ for every object n, since visiting state H implies that the content is already located. Similarly, $\tau_2(n) = 1 + m_2(n)\,\tau_3(n)$. By induction on the index i of $\tau_i(n)$, it is easy to verify that

$$\tau_0(n) = 2 + \sum_{i=1}^{L} \prod_{j=1}^{i} m_j(n)\,.$$

In essence, $\tau_0(n)$ serves to count the expected number of steps it takes to locate the object in the hierarchy of caches. However, due to the presence of the additional state H, the real number of steps is always off by one from what $\tau_0(n)$ counts. Representing the actual expected value by $\tau(n)$, therefore, $\tau(n) = \tau_0(n) - 1$ and Equation (4) follows. □

A slightly modified version of the foregoing result has also been used in [5] as a measure of "virtual round-trip time" to access contents of various popularity classes.

3.2 Optimal Cache Allocation in Tree Structure

Definition 1. (The optimal cache allocation problem) *Given a fixed total cache budget C, find the optimal breakdown of the caching budget across different levels of the tree that minimizes the overall expected time to access subject to a given content popularity profile $q(\cdot)$.*

Under the IRM assumption, for a k-ary tree with L cache levels, we formulate this problem as a non-linear integer programming as follows:

$$\begin{aligned}\boldsymbol{c}^* = \operatorname*{argmin}_{\boldsymbol{c}} & \sum_{n=1}^{N} q(n)\,\tau(n;\boldsymbol{c}) \\ \text{s.t.} \quad & \sum_{\ell=1}^{L} c(\ell)\,k^{(L-\ell)} = C\,, \quad \text{and} \\ & c(\ell) \geq 0 \text{ and integer } \forall \ell \in \{1,\ldots,L\}\,,\end{aligned} \tag{5}$$

where $\boldsymbol{c}^* \in \mathbb{N}^L$ is the vector of optimal cache sizes on the tree in which $c^*(\ell)$ denotes the optimal capacity of an *individual* cache at the ℓ^{th} level.

3.3 Numerical Results

We collected some numerical results on the problem described above utilizing the active-set algorithm of the optimization toolbox in MATLAB. As the underlying topology, we considered k-ary tree structures of depth 7. The requesters are the leaves (*i.e.*, level 0) and the source (storing a permanent copy of all objects) is at the root (*i.e.*, level 6). The 5 intermediate levels—which we call ℓ_1 to ℓ_5 caches—are cache routers with LRU replacement policy.

All named objects in the system are ranked based on their global popularity—*i.e.*, the overall frequency of requests for that object throughout the system. For these simulations, we used 1 million data objects of all the same size whose popularities follow a Zipf distribution with exponent 1. References to these objects are Poisson distributed with rates proportional to their popularities. For the time being, we make sure that the references conform to the IRM assumption and that identical objects have the same popularity among all users. We shall later explain how the IRM assumption can be relaxed by focusing on more general classes of traffic with non-stationary behavior in time and space.

Figure 2 demonstrates the optimal breakdown of the caching budget across various levels of the tree hierarchy for complete trees of degree 2 to 5. Bars show what fraction of the caching budget is allocated to various levels for any given total budget. The darker the color, the lower the cache level in the hierarchy. As seen, there is a drift towards the edge as the caching budget increases. This is trivially expected when ETTA is the objective criterion for the optimization problem. For trees with lower degrees, this behavior is more

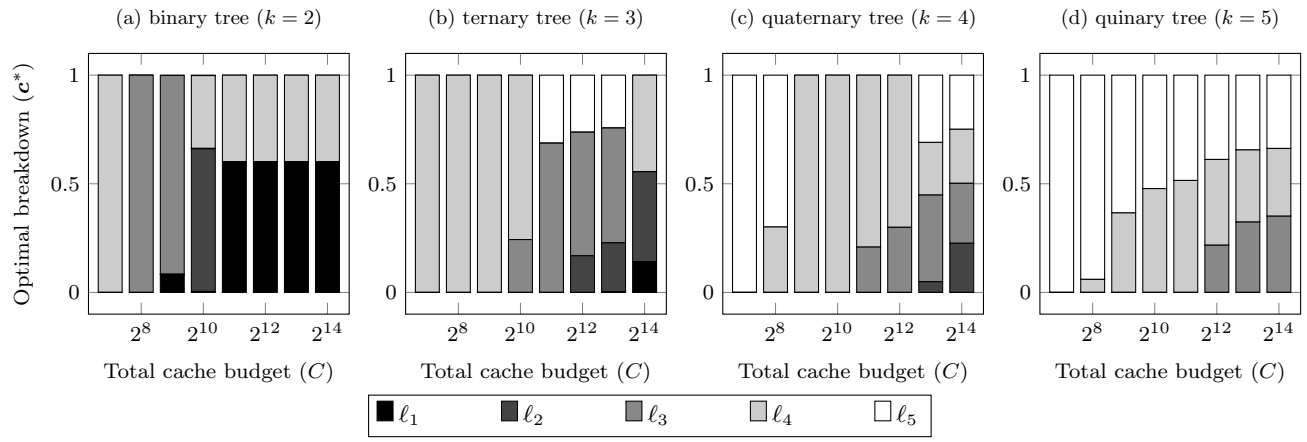

Figure 2: Optimal breakdown of caching budget across various levels of the tree for the given total cache budgets

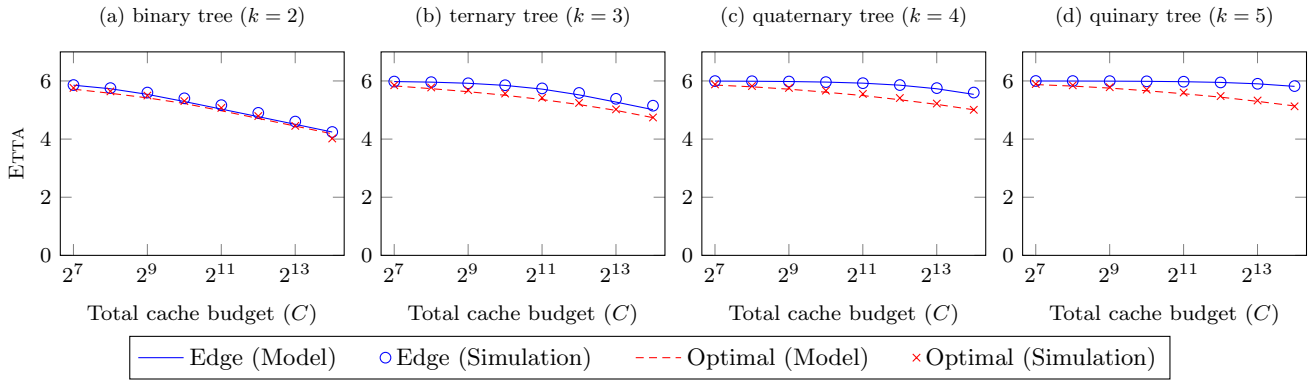

Figure 3: The overall expected time to access (ETTA) with respect to the total caching budget for optimal- vs. edge-caching

evident, because for the same total budget, edge caches in lower degree trees can receive larger shares, making edge-caching even more effective.

When the caching budget is large, edge-caching is clearly the optimal strategy. At the limit, a budget of $N \times k^L$ can be broken up evenly across all ℓ_1 caches giving each of them enough capacity to store a copy of all objects in the system. This results in an ETTA of close to 1 in the long run. The available budget, nonetheless, is usually much less than this in practice. Figure 3 serves to shed some light on the question of how different optimal caching performs in general as compared with pure edge-caching by comparing the overall ETTA for edge- with optimal caching.

The solid lines in Figure 3 represent edge-caching, where all the caching budget is evenly split across ℓ_1 caches. The dashed lines illustrate the optimal caching with a budget breakdown specified in the corresponding part of Figure 2. To verify the accuracy of these results, we also designed a discrete-event simulation based on ndnSIM [1], a NS-3 module implementing Named Data Networking. As seen, results from discrete-event simulations demonstrate almost perfect agreement with the proposed model.

Interestingly, Figure 3 suggests that edge-caching can perform comparably close to the optimal caching in practice. According to our results, the maximum difference observed between the two schemes is around 10%. The difference is seen to increase slightly with the degree of the tree. However, as Figure 2 also illustrates, the optimal breakdown tends towards the edge with an increased caching budget. This essentially means that the difference between the edge- and optimal caching is reduced with further increasing the budget. On the other hand, when the available budget is small, both edge- and optimal caching strategies appear to be equally ineffective. Thus, the maximum observed gap applies only to the cases where the available budget is modest—that is, neither so large to make edge-caching effectively optimal, nor so small to undermine the effectiveness of caching altogether.

Implementing edge-caching is practically more convenient, in that it only requires deploying ℓ_1 caches at the AS-level without any need to manipulate central routers deep in the core of the network. Although the effectiveness of edge-caching has previously been shown through extensive empirical studies [13], our work, to the best of our knowledge, presents the first formal framework as a basis to compare the two paradigms in more depth.

4. CAPTURING REFERENCE LOCALITY

To obtain useful insights out of the foregoing analysis, it is imperative to evaluate the model under realistic conditions. As discussed, the IRM assumption overlooks the correlation present among subsequent object references occurring over a certain period of time (*i.e.*, temporal locality) and/or a specific region in space (*i.e.*, spatial locality). We intro-

duce a convenient model to generate object references while preserving their spatio-temporal locality properties. Before proceeding, let us have a closer look at the intuitive interpretation of the concepts of spatial and temporal locality.

Spatial locality of reference captures the impact that the geographical diversity of the users has on the observed trace of requested objects by them. More precisely, the requests coming from a specific region in space are more likely to be similar than those collected over regions far apart. For example, a certain news object might be of special interest in a certain area, while its global impact in the geography of interest remains limited. On the other hand, globally popular objects are seen to be requested from a wider range of geographical regions.

Temporal locality of reference captures the effect that, if an object is requested at a certain point in time, more likely it will be requested again in near future. In fact, nor are the object references scattered randomly and independently over time; rather, an object might be of particular interest at a certain time interval, while its popularity gradually fades out.

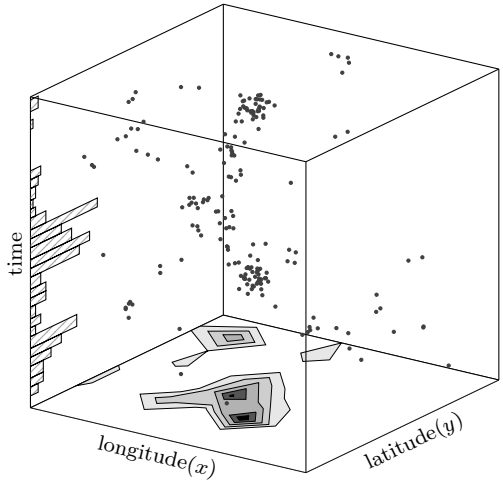

Figure 4: A cluster process representing references to a specific object file. The projection of points over X-Y plane represents the spatial density of requests, whereas the projection along the time axis reflects the temporal distribution.

4.1 Using Cluster Point Processes

In the light of the above discussion, an intuitive approach for simulating the spatio-temporal locality of object references is using "cluster point processes" [9]. A generic method for producing one such process works as follows. First, a point process $\mathbf{\Pi}$ generates "centers" of the process. Next, a point process $\boldsymbol{X}_p$ for each $p \in \mathbf{\Pi}$ produces the off-springs. The combination of these points $\boldsymbol{X} = \cup_{p \in \mathbf{\Pi}} \boldsymbol{X}_p$ constitutes a cluster process. Particularly, $\boldsymbol{X}$ is called a "Poisson cluster process" if $\mathbf{\Pi}$ is a Poisson process.

A specific example of the Poisson cluster process is "Hawkes process" [19] that is generated as follows. First, a Poisson process on $\mathbb{R}^d$ with intensity function $\rho(\cdot)$ creates the cluster centers $\mathbf{\Pi}$. Then, for each cluster center $p \in \mathbf{\Pi}$, the first-generation off-springs are generated as a Poisson process of intensity $\varphi(x-p)$, where $\varphi(\cdot)$ is a positive function on $\mathbb{R}^d$. This process continues repeatedly such that for every first-generation off-spring p_1, a Poisson process of intensity $\varphi(x-p_1)$ generates the second generation off-springs and so on. The mean number of off-springs for each center point is determined as $\beta = \int \varphi(x)\,\mathrm{d}x$. A natural requirement for this process to stop demands $\beta < 1$. Figure 4 illustrates one realization of the Hawkes process in $\mathbb{R}^3$. The contour plot on the X-Y plane represents the spatial density of the requests for a certain object, while the histogram along the time axis shows the temporal evolution of the object popularity.

Algorithm 1 Method for generating object references with localization in a d-dimensional space

Input: Number of objects (N), Zipf parameter (α) and localization factor (β)
Output: An aggregate Poisson cluster process X
Ensure: X is Zipf distributed with parameter α
1: **procedure** GENERATE-TRACE(N, α, β)
2: $X \leftarrow \varnothing$
3: **for** n from 1 to N **do**
4: $q_n \leftarrow M \times n^{-\alpha}$ ▷ M is some constant multiplier
5: $\Pi_n \leftarrow$ HAWKES-PROCESS(q_n, β)
6: $X \leftarrow X \cup \Pi_n$
7: **end for**
8: **return** X
9: **end procedure**

Input: Intensity of cluster centers (ρ) and the expected number of off-springs (β)
Output: A Poisson cluster process Π
Require: $\rho \geq 0$ and $0 \leq \beta < 1$
10: **procedure** HAWKES-PROCESS(ρ, β)
11: $n_t \leftarrow$ POISSON(ρ)
12: **for** i from 1 to n_t **do**
13: $\Pi(i) \leftarrow$ UNIFORM(0,1)
14: **end for**
15: $idx \leftarrow 1$, $end \leftarrow n_t$
16: **while** $idx < n_t$ **do**
17: $n_c \leftarrow$ POISSON(β)
18: **for** j from 1 to n_c **do**
19: $\Pi(++end) \leftarrow \Pi(idx) +$ NORMAL(0,σ)
20: **end for**
21: $n_t \leftarrow n_t + n_c$, $idx{+}{+}$
22: **end while**
23: **return** Π
24: **end procedure**

This procedure can be repeated for the number of objects in the system to generate a collective trace of all references. The procedure GENERATE-TRACE in Algorithm 1 shows the pseudocode for this with inputs N denoting the total number of objects, α as the parameter of the Zipf distribution for object popularity, and β specifying the localization factor. For an object n, Line 4 calculates the intensity q_n at which the references to that object should be generated. To ensure that the global object popularity profile follows the desired Zipf distribution, we choose this rate to be directly proportional to the global popularity of the object in the system. Consequently, references to more popular objects will be placed over a wider geography and a longer course of time, as opposed to the less popular items which may only be requested from a specific region and a certain period. The multiplier M in Line 4 is a positive constant which can be interpreted as the maximum intensity—*i.e.*, the desired intensity for the most popular (first rank) object. Depending on the choice of N and α, one may need to set the value of this multiplier sufficiently large to ensure that the lower rank objects at the tail of the popularity distribution will also have a reasonable chance to appear in the trace.

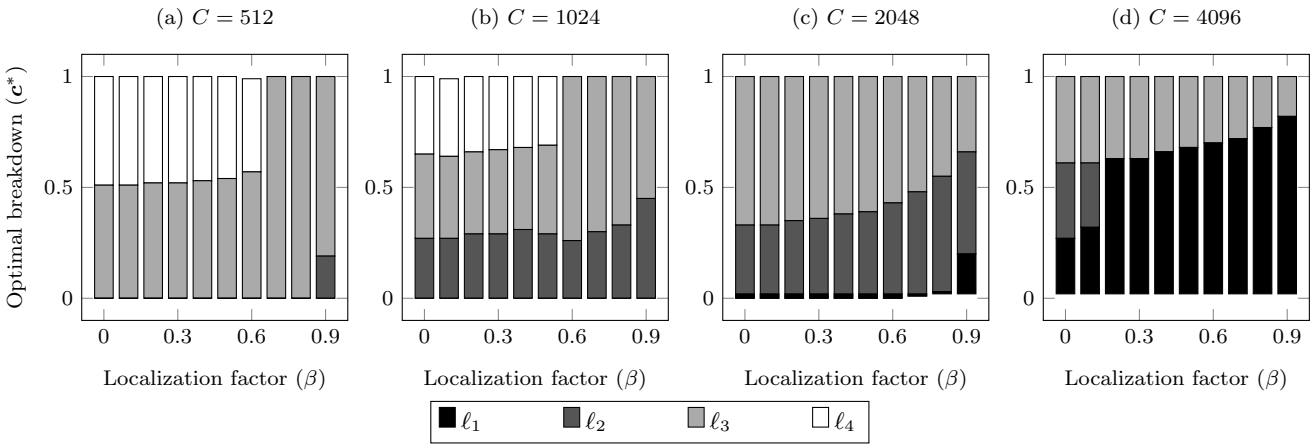

Figure 5: Optimal breakdown of caching budget across various levels of the tree for various localization factors

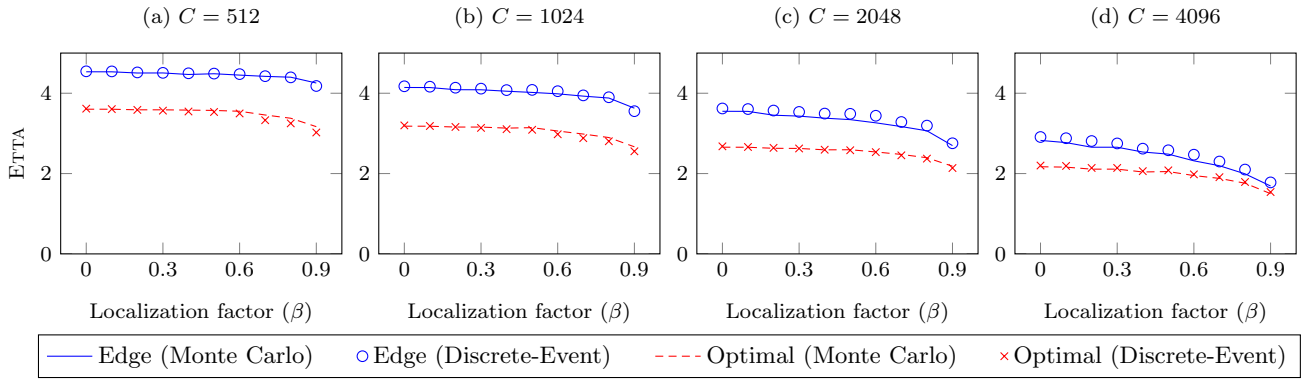

Figure 6: The overall expected time to access (ETTA) with respect to the localization factor for optimal- vs. edge-caching

Once the reference intensity is determined, a call to procedure HAWKES-PROCESS is made at Line 10 to produce the actual trace of references. The parameters ρ and β respectively determine the intensity of the centers and the expected number of next-generation off-springs in the underlying cluster process. The centers are uniformly scattered throughout the region, and for each center, the off-springs are normally distributed around it. The procedures UNIFORM and NORMAL are assumed to return coordinates in d-dimensional space with corresponding distributions and the parameters specified.

4.2 Caching under Non-stationary References

With a non-stationary stream of references, the popularity profile of objects in the system varies over both space and time. Consequently, the model discussed in Section 3 can no longer be used to analyze the behavior of the caching system. A useful insight which can help remedy this limitation is that while, in the big picture, the underlying process of object references is dynamic, when studied at a finer granularity, it can be well-approximated as a "piecewise" stationary process. In other words, if we look at the process of references through a sufficiently small window in the time-space domain, the subprocess observed exhibits a rather stationary behavior.

Let $q_u(n,t)$ be the popularity of the n^{th} object observed by a cache node u around a particular time t. As a generalization of Equation (3), we can compute this quantity as follows:

$$q_u(n,t) = \begin{cases} \lim\limits_{\Delta t \to 0} \dfrac{\mathbb{E}[N_u(n, t+\Delta t]}{\Delta t} & u \in \{\ell_1\}\,, \\ \sum\limits_{c \in \mathcal{C}_u} q_c(n,t)\, m_c(n,t) & u \notin \{\ell_1\}\,. \end{cases} \tag{6}$$

Here, $\{\ell_1\}$ denotes the set of ℓ_1 caches; $N_u(n, t+\Delta t)$ denotes the number of references to object n coming at node u during interval $(t, t+\Delta t)$; and $\mathcal{C}_u$ represents the set of caches which have u as their upstream node. In other words, the aggregate miss streams of nodes in $\mathcal{C}_u$ form the input stream of u.

In most scenarios, it is neither practical nor necessary to work with the infinitesimal limit given in Equation (6). Rather, the input stream of a ℓ_1 cache could be partitioned into a number of smaller time bins over which the input process is assumed to be stationary. The size of the time bins does indeed depend on the degree of reference locality. The more localized the input stream, the more clustered are the occurrences of the references over time; hence, smaller time bins will be required to mitigate the approximation error.

This treatment can be used in conjunction with CHE-APRX when dealing with non-stationary streams of references. The miss rates and the corresponding ETTA's can be computed separately for individual intervals. The overall ETTA, subsequently, can be calculated as the time-average of individually computed ETTA's over specific intervals.

To evaluate the accuracy of this method, we conduct some Monte Carlo simulations backed by a series of discrete-event simulations we perform in ndnSIM [1]. The underlying topology we consider is a complete tree of degree 4 and depth 6 with 4 layers of intermediate caches. Our object catalogue contains 100 files with a Zipf popularity profile of parameter 1. Algorithm 1 is used to generate a 2-D trace of object references with various degrees of localization. Leaves of the tree span across one dimension and object references are directed at their L_1-closest cache. The other dimension captures the temporal distribution of references as discussed before.

Figure 5 demonstrates the optimal breakdown of cache budget across levels of the tree for various degrees of reference locality. Again, a drift towards the edge can be observed as the available caching budget increases. This transition is further accelerated with a larger localization factor. Figure 5(d) reflects this phenomenon more vividly. In particular, we observe that an increased localization factor from 0.0 to 0.9 has almost the same impact on the performance of the caching hierarchy as doubling the caching budget does.

Figure 6 compares the overall expected time to access for edge- vs. optimal caching with the same configurations as plots in Figure 5. As discussed earlier, for numerical analysis, we split the time into smaller non-overlapping intervals (bins). With zero localization, references are generated independently. The generated trace, therefore, conforms to the IRM assumption and hence, one single time bin is considered. With a localization factor of 0.9, we found that 5 time bins yield a good approximation with a maximum error of 6% over a wide range of configurations. For other cases in between, the number of bins are chosen proportionally. At this stage, we do not know how the number of bins should be chosen optimally to minimize the approximation error. Answering this question requires a deeper understanding of the behavior of the underlying point process, and we leave this as a subject for future research.

Figure 6 also captures how fast optimal on-path caching converges towards the edge as both the caching budget and localization factor increase. In the examples depicted, the maximum difference between the two schemes is around 35%. This is reduced to 8% on the far right of Figure 6(d) where every ℓ_1 cache gets enough capacity to store 16 objects.

The main focus of our study so far was primarily on a well-defined hierarchy of caches. In what follows, we broaden the scope of our findings by considering a more arbitrary topology of caches.

5. CACHING ON RANDOM NETWORKS

Let Π_0 be a point process on d-dimensional space representing a random deployment of cache nodes. We define the local cache of a sub-region in the space of interest as follows.

Definition 2. *A node $x \in \Pi_0$ is said to be the local cache for the region $\mathcal{C}_x$ defined as:*

$$\mathcal{C}_x = \left\{ y \in \mathbb{R}^d : \|y - x\| \leq \inf_{x' \in \Pi_0, x' \neq x} \|y - x'\| \right\}.$$

In fact, $\mathcal{C}_x$ comprises the closed set of points that are geographically closer to x than any other point in Π_0. In this sense, the process Π_0 forms a Voronoi tessellation of the space similar to the construction depicted in Figure 7. The solid dots are the points generated by Π_0 and the polygons

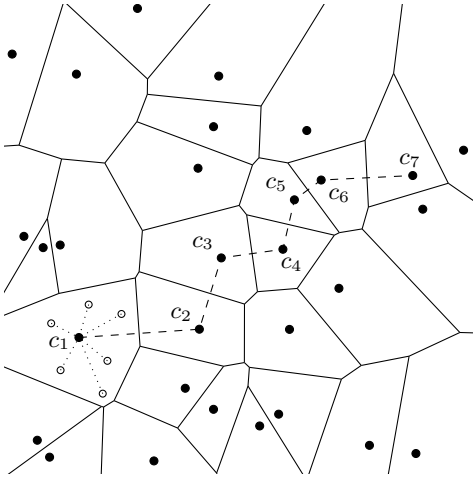

Figure 7: A Voronoi tessellation of the terrain via randomly deployed cache nodes. Solid dots are local caches to the cells they belong to, and empty dots indicate requests directed to them (shown only for one cell for clarity). The dashed line represents a path from cell 1 to cell 7. Cache node c_7 is the original source of the content to be routed to c_1, the initial requesting cell. On this way, if the content is cached at every cache node $c_i, i = 1, \ldots, 6$, it is called on-path caching. If the content is cached only at c_1, we call it edge-caching.

where they reside are the Voronoi cells. We shall refer to $\mathcal{C}_x$ as the cell of node x, hereinafter.

Each cache node is equipped with two types of storage. One part is the permanent storage used for publishing content. The other part is used for caching content from other nodes while the cache node serves to route the content towards some end-user/subscriber.

Subscription requests (generated by Algorithm 1), form a second point process. A request originating from cell $\mathcal{C}_x$ is first forwarded to x, the local cache of that cell. If x happens to have a copy of the requested object, it serves the request locally. Otherwise, it forwards the request towards the original owner of the content in a multi-hop fashion.

The connectivity among cache nodes is defined based on Euclidean distance such that every $x_i, x_j \in \Pi_0$ are connected through a bidirectional link iff $\|x_i - x_j\| \leq r$ for some constant $r > 0$. Such a paradigm is often adopted for modeling of wireless ad hoc networks and might not well represent a typical wired topology. Still, we believe it is interesting to study the performance of various caching schemes on a more general and irregular type of topology such as that of a random geometric graph.

5.1 The Routing and Caching Process

The routing is performed along the shortest path connecting source-destination pairs. For simplicity, we choose the critical radius r large enough to ensure that the network is connected. Hence, there always exists at least one path connecting each subscribing cell to the publishing source. Such paths typically cross through several cells and the traffic carried along them may be cached at the local caches of the cells where they intersect.

On the way towards the source, if a valid replica of the requested object is located at any of the intermediate caches along the path, the request will be handled locally and it will not be forwarded beyond that point. However, if no cache

hit occurs along the path, the request will be served by the original source and the requested object will be routed back towards the requester on the reverse path.

For a well-structured tree topology, we observed suggestive evidence that edge-caching can perform comparably close to optimal caching in certain situations. Implementing optimal caching on a random configuration, nevertheless, is challenging if not impossible at all. This is perhaps why many existing ICN proposals adopt a simplified version of on-path caching in which all routers blindly cache every piece of information they relay.

Implementing edge-caching, in contrast, is not much of a burden on a random network so long as a clear definition of "edge" is given. Since object requests are scattered all over the network, there is no physical boundary to separate edge from the rest of the network. Instead, we give a logical definition of the edge. In fact, we say edge-caching takes place if caching is only performed at the local cache of the last cell where the traffic is being served,—*i.e.*, the destination cell. According to this definition, a cache router is an edge cache for the cell it resides in and a non-edge cache for the traffic it relays to all other cells. An interesting aspect to investigate is a performance comparison of the simplified on-path caching—which for brevity we shall refer to as "on-path"—versus edge-caching in the above described configuration.

Although appearing different, the system we just described has many characteristics in common with the hierarchical topology we discussed in the previous section. In particular, for any given object, the original publisher (source) serves as the root of a tree. The object requesters (*i.e.*, destinations) are the leaves, which can be from anywhere within the network. Of course, the induced tree structures differ for various object files. Consequently, a cache node can be part of several such logical trees and at different levels.

5.2 Simulation Results

We perform event-driven simulations on ndnSIM [1] to compare the performance of caching at the edge versus the standard on-path caching on a random geometric topology. The network consists of 200 cache nodes distributed uniformly and at random over a region of 100×100 square units. Nodes' radio range is set to 12 units giving each node an average degree of 8.93. We use a total of 1000 content objects with a Zipf popularity distribution of parameter 1. The objects are also uniformly distributed among nodes which act as original publishers of the designated objects. Hence, some nodes may publish more than one data object while some others none. With the foregoing settings, the following measurements are performed in the steady-state of the system when all caches are full.

Figure 8 shows how the average hop-count decreases with an increased locality factor when each node has a caching storage of size 10. In this case, edge-caching outperforms the standard on-path caching even under the IRM assumption (*i.e.*, $\beta = 0$). This behavior is rather surprising because when the references are generated independently, there should seemingly exist no difference between the two schemes in terms of cache hits. However, a subtle observation is that replacements generally take place at a higher rate with on-path caching than with the edge-caching. This is due to the replacements that a cache incurs while relaying traffic to other cells. These replacements do not take place when caching is only done for the edge traffic.

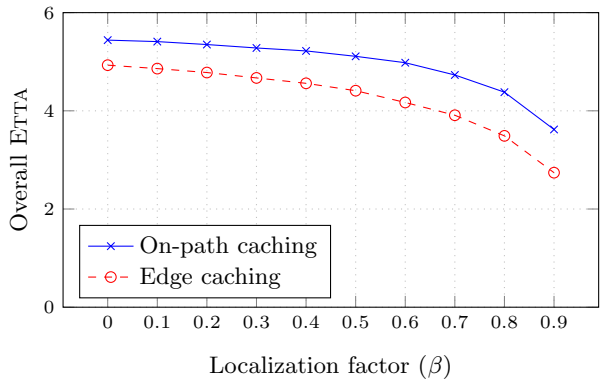

Figure 8: Overall expected time to access an object in a random geometric topology for various degrees of reference locality (from IRM ($\beta = 0$) to highly localized ($\beta = 0.9$))

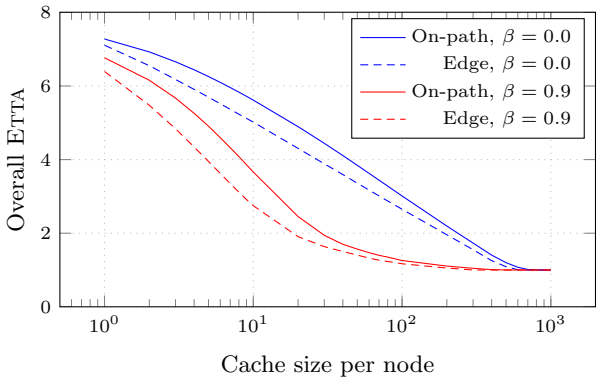

Figure 9: The impact of increasing the cache size on the caching gain for various degrees of reference locality. Edge-caching outperforms on-path caching in almost all scenarios.

Not all such replacements are useful. In fact, with a Zipf-like distribution, a vast majority of objects at the tail are individually unpopular and very unlikely to be requested. Yet, in the big picture, it is much more likely to see *some* object from this whole population of less popular items being referenced by *some* node throughout the network. With on-path caching, all such references result in replacements along the entire path serving the traffic to the destination cell. Once referenced, however, because the object is of little global interest, the odds are small that any of these affected caches serve any subsequent reference to the same object in near future. The replaced item takes up a space that could have otherwise been dedicated to a more popular item and thereby, diminishes the caching gain.

With edge-caching, such useless replacements occur at a lower rate and the caching capacity is utilized more efficiently. The same arguments hold in case of higher degrees of locality resulting in an even sharper contrast.

Figure 9 illustrates the impact of increasing the caching budget on the average hop-count. As seen, edge-caching outperforms on-path caching for all cache sizes and over various degrees of locality. The enhancements attained through increasing the budget size become more pronounced with a higher degree of reference locality. Another observation is that a localization factor of 0.9 requires roughly 6 times less caching budget to yield the same overall Etta than it does under the independent reference model.

6. CONCLUSIONS AND FUTURE WORK

A computational framework was presented to compare the performance of in-network caching mechanisms in ICN. In particular, we compared optimal on-path caching against the simple strategy of caching only at routers near the consumers of an ICN in terms of their overall expected time to accessing content objects. The results using the commonly-used independent reference model showed that while the optimal breakdown of caching budget is markedly influenced by factors such as topology and caching budget, optimal caching provides only marginal benefits over edge-caching in most scenarios. We investigated the impact of locality of reference on the performance of ICNs, and introduced a tool to synthesize traces of object requests that preserves their spatial and temporal locality properties. The results using this model demonstrate that, while optimal caching naturally tends towards the edge with an increased caching budget, higher degrees of reference locality further accelerate this transition. This suggests that the difference between edge- and optimal on-path caching is far less than what the existing models based on the IRM assumption predict.

We also compared the on-path caching approach assumed in most ICN architectures today against edge-caching using random geometric graphs to model ICNs of irregular arbitrary topologies. The results of simulations in ndnSIM [1] confirm the result in our models, and in fact show that edge-caching outperforms on-path caching.

The results of this work open up new avenues for research in ICN. New ICN architectures should be investigated that exploit content routing with edge-caching. It is important to broaden the scope of this study by considering more realistic types of topologies than we have used, and the locality model we introduced should be fit against real-world traces to examine how localized content requests are in actual networks.

Acknowledgments

The authors gratefully thank the anonymous reviewers for their constructive comments on an earlier version of this manuscript. This research was sponsored in part by the Jack Baskin Chair of Computer Engineering at UCSC.

7. REFERENCES

[1] A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN, 2012.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A Survey of Information-Centric Networking. *IEEE Commun. Mag.*, 50(7):26–36, 2012.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. IEEE INFOCOM*, pages 126–134, 1999.

[4] G. Carofiglio, M. Gallo, and L. Muscariello. Bandwidth and Storage Sharing Performance in Information Centric Networking. In *Proc. ACM SIGCOMM Workshop on ICN*, pages 26–31, 2011.

[5] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino. Modeling Data Transfer in Content-Centric Networking. In *Proc. ITC*, pages 111–118, 2011.

[6] W. Chai, D. He, I. Psaras, and G. Pavlou. Cache "Less for More" in Information-Centric Networks (Extended Version). *Comput. Commun.*, 36(7):758–770, 2013.

[7] H. Che, Y. Tung, and Z. Wang. Hierarchical Web Caching Systems: Modeling, Design and Experimental Results. *IEEE J. Sel. Areas Commun.*, 20(7):1305–1314, 2002.

[8] L. Cherkasova and M. Gupta. Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change. *IEEE/ACM Trans. Netw.*, 12(5):781–794, 2004.

[9] D. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*, volume I: Elementary Theory and Methods. Springer, New York, NY, USA, second edition, 2003.

[10] A. Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. *ACM SIGMETRICS Perform. Eval. Rev.*, 18(1):143–152, 1990.

[11] P. Danzig, R. Hall, and M. Schwartz. A Case for Caching File Objects Inside Internetworks. *SIGCOMM Comput. Commun. Rev.*, 23(4):239–248, 1993.

[12] G. Domingues, E. Silva, R. Leão, and D. Menasché. Enabling Information Centric Networks through Opportunistic Search, Routing and Caching. arXiv:1310.8258 [cs.NI], 2013.

[13] S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. *ACM SIGCOMM Comput. Commun. Rev.*, pages 147–158, 2013.

[14] R. Fonseca, V. Almeida, M. Crovella, and B. Abrahao. On the Intrinsic Locality Properties of Web Reference Streams. In *Proc. IEEE INFOCOM*, pages 448–458, 2003.

[15] C. Fricker, P. Robert, and J. Roberts. A Versatile and Accurate Approximation for LRU Cache Performance. In *Proc. ITC*, pages 1–8, 2012.

[16] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance Evaluation of the Random Replacement Policy for Networks of Caches. *Perform. Evaluation*, 72(0):16–36, 2014.

[17] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-Centric Networking: Seeing the Forest for the Trees. In *Proc. ACM HotNets*, pages 1–6, 2011.

[18] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube Traffic Characterization: A View from the Edge. In *Proc. ACM IMC*, pages 15–28, 2007.

[19] A. Hawkes. Spectra of Some Self-Exciting and Mutually Exciting Point Processes. *Biometrika*, 58(1):83–90, 1971.

[20] P. Jelenković and A. Radovanović. Least-Recently-Used Caching with Dependent Requests. *Theor. Comput. Sci.*, 326(1):293–327, 2004.

[21] J. Kurose. Information-Centric Networking: The Evolution from Circuits to Packets to Content. *Comput. Netw.*, 66(0):112–120, 2014.

[22] V. Martina, M. Garetto, and E. Leonardi. A Unified Approach to the Performance Analysis of Caching Systems. In *Proc. IEEE INFOCOM*, 2014.

[23] N. Melazzi, G. Bianchi, A. Caponi, and A. Detti. A General, Tractable and Accurate Model for a Cascade of LRU Caches. *IEEE Commun. Lett.*, 18(5):877–880, 2014.

[24] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic In-Network Caching for Information-Centric Networks. In *Proc. ACM SIGCOMM Workshop on ICN*, pages 55–60, 2012.

[25] I. Psaras, R. Clegg, R. Landa, W. Chai, and G. Pavlou. Modelling and Evaluation of CCN-Caching Trees. In *IFIP Networking*, pages 78–91. Springer, 2011.

[26] J. Roberts and N. Sbihi. Exploring the Memory-Bandwidth Tradeoff in an Information-Centric Network. In *Proc. ITC*, pages 1–9, 2013.

[27] P. Rodriguez, C. Spanner, and E. Biersack. Analysis of Web Caching Architectures: Hierarchical and Distributed Caching. *IEEE/ACM Trans. Netw.*, 9(4):404–418, 2001.

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

[29] E. Rosensweig, D. Menasche, and J. Kurose. On the Steady-State of Cache Networks. In *Proc. IEEE INFOCOM*, pages 863–871, 2013.

[30] D. Rossi and G. Rossini. Caching Performance of Content Centric Networks Under Multi-Path Routing (and More). Technical report, Telecom ParisTech, 2011.

[31] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal Locality in Today's Content Caching: Why It Matters and How to Model It. *ACM SIGCOMM Comput. Commun. Rev.*, 43(5):5–12, 2013.

[32] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel. A Trace-Driven Analysis of Caching in Content-Centric Networks. In *Proc. ICCCN*, pages 1–7, 2012.

[33] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie. Optimal Cache Allocation for Content-Centric Networking. In *Proc. IEEE ICNP*, pages 1–10, 2013.