Approximate Models for General Cache Networks

Elisha J. Rosensweig
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003–9264
Email: elisha@cs.umass.edu

Jim Kurose
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003–9264
Email: kurose@cs.umass.edu

Don Towsley
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003–9264
Email: towsley@cs.umass.edu

***Abstract*—Many systems employ caches to improve performance. While isolated caches have been studied in-depth, multi-cache systems are not well understood, especially in networks with arbitrary topologies. In order to gain insight into and manage these systems, a low-complexity algorithm for approximating their behavior is required. We propose a new algorithm, termed *a-NET*, that approximates the behavior of multi-cache networks by leveraging existing approximation algorithms for isolated LRU caches. We demonstrate the utility of *a-NET* using both per-cache and network-wide performance measures. We also perform factor analysis of the approximation error to identify system parameters that determine the precision of *a-NET*.**

I. Introduction

Many systems employ caching as a means to reduce the load on access links and shorten access time to selected content. While the basic caching unit is a single cache, equipped with management policies, some systems make use of several caches linked together, allowing each cache to also forward requests to its neighbors when needed. Common examples of such systems are hierarchical web and file system caches. Recently [1] [2] [3], there have been proposals for multi-cache designs over networks of Internet-like scale and structure.

Determining the performance of a specific multi-cache system is extremely difficult. Even for the single, isolated cache using the popular LRU replacement policy, the complexity of exact models of cache contents and performance grow exponentially as a function of cache size and the number of files in the system [4], making them ineffective as exact modeling tools. For this reason, a more useful approach is to *approximate* the behavior of the caching system, allowing some measure of inaccuracy in return for simpler modeling techniques. This has been done for isolated caches and for some cache networks, namely cache hierarchies [5].

The main drawback of existing models for networked caches is their limited scope. They address hierarchical topologies (i.e., trees), in which the source of content is connected to a node at the top of the hierarchy (i.e., the root of the tree), and rely heavily on this structure when constructing the approximation. Furthermore, due to the complexity of these systems, the models are developed for small topologies (e.g., 2-level trees), and do not easily scale up for use in larger topologies. These limitations make existing solutions inapplicable when dealing with arbitrary topologies or large-scale cache networks. Instead, what is needed is an approach that can be applied to any topology.

In this paper we present *a-NET*, an algorithm for approximating the rate of incoming and miss streams for each file at every node in a network using LRU caches. We refer to the output of *a-NET* as a **m**ulti-**c**ache **a**pproximation, or MCA for short. The approach taken by *a-NET* is to decompose the problem and compute a **s**ingle-**c**ache **a**pproximation (SCA) for each individual cache in the network, where the request misses for each file at each cache are forwarded towards a content source. The incoming request stream at each cache is thus reevaluated as a combination of both exogenous requests as well as portions of the miss stream of each cache's neighbors. *a-NET* is an iterative process, updating the incoming request stream at each cache and recomputing its miss stream (which then becomes part of the input stream to neighboring caches) using an SCA generating algorithm, until the entire network converges to a fixed point. Unlike existing models, that can generate MCAs for specific topologies, *a-NET* can compute an MCA for any topology, regardless of structure or scale, as long as the routing tables remain constant.

The contributions of this paper are:

- We develop *a-NET*, a novel MCA generating algorithm for general-topology cache networks, that utilizes the SCA algorithm described in [4] for approximating LRU caches.
- We demonstrate the behavior of *a-NET* for multiple topologies, and identify key parameters that affect its performance. Specifically, we show that dependencies within the reference stream are the main cause of inaccuracy for *a-NET*, and that an increase in network connectivity can greatly reduce this problem.
- We construct a Markov model that expresses the inter-request distances in a cache miss stream for a range of arrival distributions. Since the miss stream of one cache becomes part of the incoming stream of its neighbors, we use this model to demonstrate the effects of non-IRM miss streams on the hit-ratio at neighboring caches.
- We evaluate the accuracy of *a-NET*, in terms of both per-cache and system-wide metrics. Individual caches are evaluated using the miss probability of each cache, while the performance of the entire system is measured in terms

This work was supported in part by the National Science Foundation under CNS-0626874 and CNS-0519922. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

of the average number of hops a request traverses till content is located.

The structure of this paper is as follows. In Section II we present *a-NET*, and motivate our topics of focus in this paper with an example of its performance. In Section III we develop a general approach for factor analysis of the prediction errors of *a-NET*. We apply this approach to tree topologies, and show how this analysis can assist in determining how *a-NET* will perform in specific settings. Next (Section IV), we analyze one of the most likely causes for approximation error in *a-NET*, namely the *IRM-violation* in the cache miss stream, using a Markov chain model to explain certain properties of the prediction error in *a-NET*. Section V presents the performance of *a-NET* over a wide range of topologies, for both cache-specific performance metrics as well as network-wide metrics. Section VI presents a survey of related work on cache approximations and cache networks, and we conclude with a summary of our findings, and future work, in Section VII.

II. Approximating Cache Networks using *a-LRU*

A. Model Description and Problem Statement

We begin by describing the system of interest in this paper, namely cache networks. Let $G = (V, E)$ be a network of caches, $V = \{v_1, ...v_n\}$, $E \subseteq V \times V$. The cache size at node $v \in V$ is denoted $|v|$. Additionally, let $F = \{f_1, ..., f_N\}$ be the set of files in the system. Each file is stored permanently at one or more servers $S = \{s_1, ..., s_m\}$ that are attached to the network, each to one or more $v \in V$. For all $s \in S$ define $files(s) \subseteq \{1, ..., N\}$ to be the file indices stored at the server, s.t. $\left|\bigcup_{s \in S} files(s)\right| = N$. Also, for all $s \in S$ and $v \in V$ let $\chi(s, v) = 1$ iff source s is attached directly to v, and otherwise $\chi(s, v) = 0$. For simplicity of presentation, we assume for the rest of the paper that each file source is attached only to a single cache in the network, denoted v_s.

In this paper we address networks with routing tables that do not change over time. For presentation purposes, we assume shortest path routing is used. Let a *path* P be an ordered set of nodes $P = (v_{P_1}, ..., v_{P_j})$ such that for all $1 \leq i < j$, $(v_{P_i}, v_{P_{i+1}}) \in E$. Given a node v and a file f_i s.t. $i \in files(s)$, let $P_i^v = (v_{P_1} = v, ..., v_{P_j} = v_s)$ be the shortest path from v to the source of f_i, v_s, where distance is measured in the number of hops. In case of a tie, one path is selected at random and is maintained hereafter.

At each node in this system, a Poisson stream of file access requests arrives exogenously. A request for f_i is denoted req_i. For all $1 \leq i \leq N$, $v \in V$, $\lambda_{i,v}$ is the rate of exogenous requests for file f_i at node v. When a request for file f_i arrives at a cache v, it generates a *hit* if the file is located at the cache and a *miss* if not. In the event of a miss, the request is forwarded to the next-hop cache along P_i^v, or to s if $\chi(s, v) = 1 \wedge i \in files(s)$. In the event of a hit, the file is forwarded along the reverse path taken by the request, and cached at each node along the way. If the cache is full, one of the files in the cache is evicted to make room for the new file. Following common practice (e.g., [4], [5], [6]), we assume that all files have the same size, and so the cache size $|v|$ can be expressed in terms of the number of files it can hold.

For a given path P_i^v, let $P_i^v[j]$ be the j-th node in the path, s.t. $P_i^v[1] = v$ and $P_i^v[2]$ is the next hop from the originating node v. Given two nodes $v, v' \in V$, define

$$R(v, v') = \{i : v' = P_i^v[2]\}.$$

This is the set of all request ids i for which v' is on the next hop from v along the shortest path to source s s.t. $i \in files(s)$. Let $r_{i,v}$ be the combined incoming rate of requests for f_i at node v, and let $m_{i,v}$ be the miss rate of f_i at node v. Then the rate of requests for f_i at node v can be expressed as

$$r_{i,v} = \lambda_{i,v} + \sum_{v':i \in R(v',v)} m_{i,v'} \tag{1}$$

Note that while $\lambda_{i,v}$ is a Poisson stream of exogenous requests, the miss stream of a cache is not, and so when we refer to $r_{i,v}$ as the incoming rate at the node we are referring to the *average* rate of requests.

The miss rates at each node depend on several factors, in addition to the cache management policies. One of these is the time it takes to load content to the cache after a cache miss. In this paper, we follow common practice [4][5], and assume that the file download time after a cache miss is significantly smaller than the inter-request. Thus, once a cache miss occurs, the file is assumed to be instantaneously downloaded into the cache.

Our goal in this paper is to develop an algorithm that can predict, with high accuracy, the incoming and miss streams at each of the caches, given the exogenous incoming rates. With such an algorithm, cache network developers can evaluate the performance of the cache network itself efficiently. To determine the incoming and miss streams, we must solve the system of equations (1), for all i and v, and this in turn requires determining the miss rate for each file at each node. To this end we develop *a-NET*, our MCA generating algorithm.

B. From a-LRU to a-NET

In [4], Dan and Towsley developed an SCA algorithm for LRU and FIFO caches. As explained shortly, *a-NET* uses the LRU SCA algorithm, denoted *a-LRU*, to compute an MCA for the entire network. We begin, therefore, with a short description of *a-LRU*.

Let $\vec{p_v} = (p_{1,v}, \ldots, p_{N,v})$ be the steady-state incoming request distribution for files in F at a certain cache v. *a-LRU* can be thought of as a function $contents(\vec{p_v}, |v|) = \vec{q_v}$, where $\vec{q_v} = (q_{1,v}, \ldots, q_{N,v})$ is the vector consisting of the probability for each file to be present in the cache at a random point in time.

a-LRU was developed for request streams that conform to the **I**ndependent **R**eference **M**odel, or IRM for short, which means that the probability that the j-th request will be req_i, given past requests, is still $p_{i,v}$. For a request stream conforming to IRM, we prove in [7] that

$$m_{i,v} = r_{i,v} \cdot (1 - q_{i,v}). \tag{2}$$

Given our discussion thus far, we can now define the MCA generating algorithm, *a-NET*, an algorithm for solving the following set of equations, for each $v \in V$ and i s.t. $f_i \in F$:

$$r_{i,v} = \lambda_{i,v} + \sum_{v':i \in R(v',v)} m_{i,v'} \quad (3)$$

$$p_{i,v} = \frac{r_{i,v}}{\sum_{j=1}^{N} r_{j,v}} \quad (4)$$

$$\vec{q_v} = contents(\vec{p_v}, |v|) \quad (5)$$

$$m_{i,v} = r_{i,v} \cdot (1 - q_{i,v}) \quad (6)$$

Eq. (3) is identical to Eq. (1), which combines the exogenous request stream with the miss streams of the neighbors of v to get the incoming request stream at each cache. Eq. (4) defines $p_{i,v}$ as the relative portion of requests for f_i at v. Eq. (5) repeats the definition of the $contents(\cdot,\cdot)$ function, and Eq. (6) is identical to Eq. (2). Note that Eq. (6) has only been established for IRM streams. When the request streams consist also of the miss streams of neighbors, that do not conform to IRM, the equation may not accurately predict the miss rate. This issue is studied in detail in Sections III and IV.

a-NET solves these equations iteratively. Initially, we set $m_{i,v} = 0$ for all i and v. Note that in the order these equations are presented, each equation relies only the values of the exogenous rates $\lambda_{i,v}$, which we assume are given, and information available from previous equations. In each iteration, *a-NET* solves equations (3) through (6) in order, for all caches, using the output of the previous iteration as input to the next. *a-NET* halts when a predefined precision threshold ϵ is met. In our implementation, we used the mean-square distance between the request rates of all files at all nodes as the halting criterion. While we have not established convergence, the algorithm converged in all cases that we tested.

C. a-NET performance - an example

To motivate the issues we discuss in the coming sections, we present here an example of the performance of *a-NET* over a simple topology. When using an isolated cache, a commonly used performance metric is the *miss probability* of the cache. Approximations for these caches are then evaluated by observing the **M**iss **P**robability **R**atio (MPR) between the predicted and actual behavior [4].

We simulated the behavior of a 10-by-10 torus cache network. The exogenous request distribution and rate is the same for all caches, and requests are distributed according to Zipf distribution with parameter 1 (i.e., the i-th most popular file has $p_i = \frac{1/i}{\sum_{j=1}^{N} 1/j}$), as has been observed in real web access traces [8]. There are $|F| = 500$ files in the system, cache sizes are $|v| = 50$, and there are 4 sources of content $s_1 - s_4$ with each file stored at exactly one source. We compared the behavior of the system to that of the approximation generated by *a-NET*, the results presented in Figure 1. The top plot shows the MPR per node. The bottom plot shows the standard deviation of the incoming file request distribution at each cache node, approximation vs. simulation.

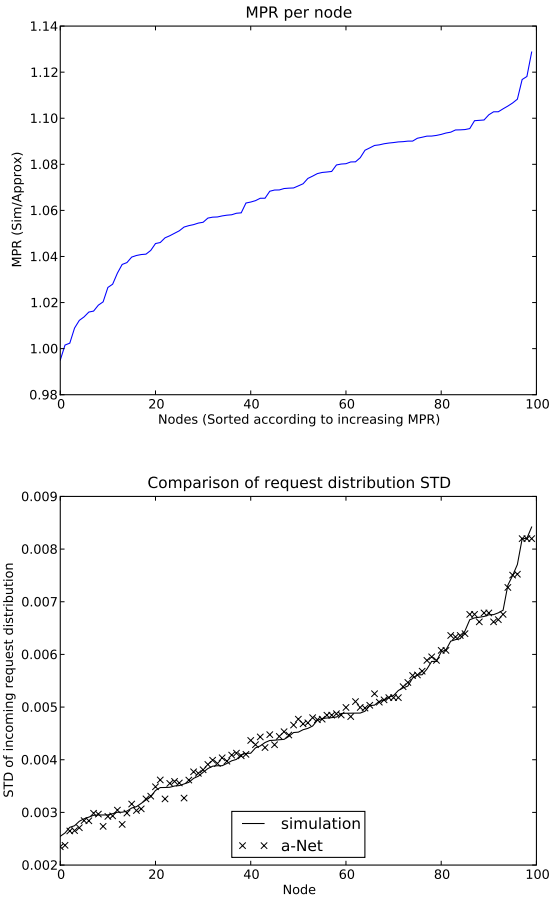

Fig. 1. *Top:* MPR for 10-by-10 torus, with file requests distributed using Zipf distribution, $|F| = 500$, and all caches of size $|v| = 50$. Nodes are sorted by increasing MPR. *Bottom:* Standard deviation of incoming request distribution at each node. The solid line (sim inc) is for STD in the simulation, while the x's (approx inc) are for the approximation. Nodes are sorted according to increasing STD in the simulation. As can be seen, our model correctly captures the trend of the STD curve.

Three main features are evident in Fig. 1. First, focusing on the MPR, *a-NET* consistently *under-estimates* the number of misses that occur at each cache. Second, the MCA produced by *a-NET* does not err by more than 13%, with a median error of 7% . These results indicate that, even though the IRM assumptions are invalid for cache networks, *a-NET* predictions are close to the actual behavior. Finally, the STD of the approximated incoming distribution is almost identical to that in the simulation. This suggests that the approximation is using as input at each node a steady-state distribution that is close to the actual distribution observed at that node, and so the cause for increased MPR must lie elsewhere. Thus, we next more closely investigate what factors cause the errors we see here, and study under what conditions are these errors minimized.

III. *a-NET* ERROR DECOMPOSITION

In this section we analyze the sources of errors in *a-NET*. Our goals here are (a) to determine the possible causes of error; (b) to show how these errors can be distinguished; and

(c) to use this information to determine in which scenarios *a-NET* will provide accurate predictions.

We consider three potential causes for *a-NET* inaccuracies:

1) **Inherent Prediction Error in the underlying SCA algorithm.** Since *a-LRU* only approximates cache behavior, even with the correct request distribution the results may differ from the actual system behavior.
2) **Violation of IRM assumption.** The violation of IRM may adversely affect the performance of *a-NET* in two ways. First, *a-LRU* was designed for and evaluated only under the IRM assumptions. Second, the miss rate of each file is calculated using Eq. (2), which may not hold for IRM streams.
3) **Input Error (or: Propagated Error).** The input given to *a-LRU* for cache v during *a-NET* execution includes outputs predicted by *a-LRU* for all of v's neighbors. Since the prediction of *a-LRU* is inaccurate, the request distribution given as input to *a-LRU* at each cache may not be the actual distribution at that cache. This can produce inaccurate predictions.

We therefore examine the contributions of each of the errors in a given scenario, comparing a simulation (denoted SIM) to the *a-NET* approximation (denoted APP). We disentangle the different errors from each other via factor analysis, extracting from the simulation results the probability of a request for each file at each node, and then evaluating the per-cache performance in two additional scenarios:

1) **Per-cache simulation with IRM traffic,** denoted SIM-IRM. Here, we simulate the behavior at each cache using the file request distribution extracted from SIM, but with file requests being generated from this distribution according to IRM.
2) **Per-cache approximation with simulation-driven traffic,** denoted APP-DRIVEN. Here we evaluate the cache performance using *a-LRU* at each cache, using the file request distribution extracted from SIM. Recall that the *a-LRU* algorithm assumes IRM arrivals.

We shall refer to these as the *pseudo-simulation* and *pseudo-approximation* respectively. In [4] the authors demonstrate that *a-LRU* gives close to optimal results for common scenarios, and so our goal here is to determine what part of the approximation error is due to IRM violation, and what part is due to input error, as defined above. We do so by comparing the predictions of *a-NET* in these different scenarios. Specifically,

- Comparing SIM to APP provides the MPR performance of *a-NET*, as in Fig. 1.
- Comparing SIM to APP-DRIVEN isolates the influence of IRM violation on the performance of *a-NET*, since the input at each cache is the same for both simulation and pseudo-approximation.
- Comparing SIM-IRM to APP-DRIVEN removes both the effects of IRM violation and input error, since the input at each cache is the same and the arrival streams at caches in SIM-IRM conform to IRM.

Factoring *a-NET*s approximation error in into its components can help develop improved prediction algorithms, that address these errors, as well as determine which cases will be more prone to prediction errors. We demonstrate this second point next for the case of cache trees.

A. Cache Trees

When using shortest path routing, cache trees form in a cache network with a single source of content. Each node forwards its entire miss stream along a single link, the one on the shortest path to the source. Leaves are therefore nodes whose input stream does not include the miss stream of any neighboring cache. For the purpose of this case study, we consider only complete k-ary trees. As in the example case from Section II, we assume that (a) all caches are of the same size, and (b) the exogenous request stream at each cache is the same. These assumptions are maintained throughout this paper.

We initially consider the case of a linked list of h caches $v_0, \ldots, v_{h-1}$, with cache v_{h-1} linked to the single source (Fig. 2). We simulated the behavior of the system with parameters $h = 10$, $|v| = 50$, $N = 500$, where the exogenous request distribution is Zipf with parameter 1. We generated, additionally, the pseudo-simulation and pseudo-approximation as described above, and plotted the pair-wise MPR between each of the simulations (standard (SIM) and pseudo (SIM-IRM)), and each of the approximations. The results of these comparisons are presented in Figure 3.

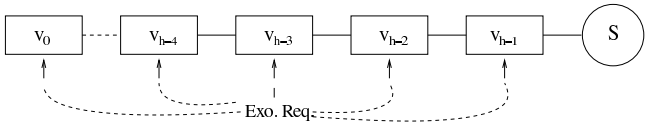

Fig. 2. k-ary tree for k=1, with request streams arriving exogenously at each node.

Several conclusions may be drawn from Figure 3. First, we note that when both the input error and IRM-violation errors are removed, the performance of the algorithms is close to optimal (a ratio of 1.0), a strong indication that there are no additional hidden causes for error. Second, as in the example from Section II, the error leads *a-NET* to consistently underestimate the probability of misses. Finally, it is clear that the input error is negligible in the case portrayed in Fig. 2, and that IRM-violation is the main source of error, which increases at caches closer to the source. This supports the observation made at the end of Section II, that *a-NET* provides a good estimate of the incoming request distribution at each node. Thus, one would expect that in scenarios where IRM is violated to a lesser degree, the performance of *a-NET* would improve.

Support for this last hypothesis can be obtained from the performance of *a-NET* for larger trees, as we increase the branch factor k of the tree. As k grows, the incoming request stream at upper-level caches become more IRM-like. To understand why, consider a node A with two child nodes B and C, each of which is the root of its own sub-tree. Requests

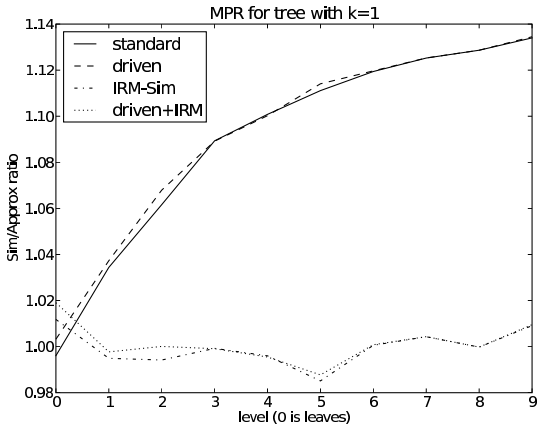

Fig. 3. Miss probability ratio for standard (SIM / APP), driven (SIM / APP-DRIVEN), IRM-Sim (SIM-IRM / APP) and driven + IRM-Sim (SIM-IRM / APP-DRIVEN). Performance is better when closer to 1.0. When IRM-violation is removed, the performance of *a-NET* at all caches is close to optimal

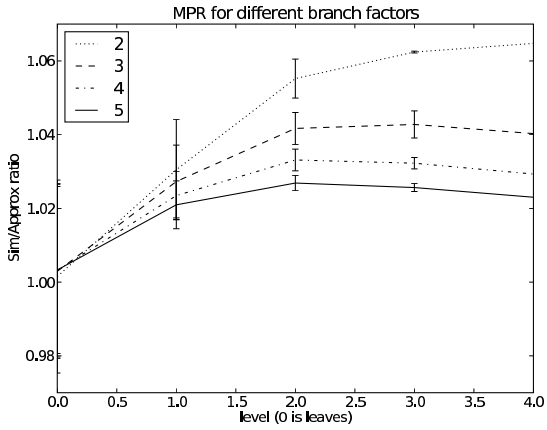

Fig. 4. Miss probability ratio for trees with branch factor $k = 2, 3, 4, 5$.

arriving at B are independent of those arriving at C since their respective sub-trees do not have any node in common. Therefore, the miss streams of B and C are independent of *each other*. However, the miss stream of B at each epoch depends on the state of cache B in that epoch. It is expected, therefore, that the aggregate of these two streams is in some sense closer to IRM than each of them would be on its own. As k goes to infinity, we get closer to a purely IRM request stream at A.

Based on this insight, we expect the performance of *a-NET* to improve as k grows. The results of testing this hypothesis for $k = 2, 3, 4, 5$ are shown in Figure 4. For each level in the tree, we calculated the MPR for each cache in that level, then plotted the mean MPR at that level with a 95% confidence interval. Our results clearly show that as the branch factor increases, so too does the accuracy of *a-NET*. We tested the error composition for additional distributions over tree topologies - uniform, truncated arithmetic and geometric - and consistently observed this behavior. When we turn our attention to general topologies, similar behavior occurs as the average node degree grows. This is demonstrated and discussed in detail in Section V.

IV. Understanding IRM Violation in Cache Networks

As we've just seen, IRM-violation can cause *a-NET* prediction errors. In this section we present insight derived from an analytical model supporting the observation that *a-NET* generally underestimates the miss probability.

Let us begin with some intuition. It has been shown (for example, see [9]) that a chain of LRU caches performs poorly. One of the reasons for this is the lack of *locality of reference* in the miss streams of caches. When a cache miss occurs at v for file f, the file is downloaded into the cache, and so in order for another cache miss to occur it first must be evicted from the cache. Until this eviction occurs, the miss stream will not see another request for this file. This causes requests for file f to be, on average, farther apart in the miss stream than they are in the incoming stream, reducing the effectiveness of next-hop caches. This behavior can also help explain why *a-NET* consistently tends to underestimate the miss probability of caches. The inter-arrival distance between requests for f_i are likely to be greater in non-IRM miss-streams than in IRM streams. Since *a-NET* predicts misses assuming IRM, the miss probability it predicts will be lower than actually exhibited.

While the behavior just described is intuitive, and has been demonstrated empirically for many scenarios, to the best of our knowledge there is little analytical support for it. In Section IV-A we present a Markov model for the simple (and tractable) case of an IRM request stream with a distribution close to uniform, as defined shortly. Next (Section IV-B), we use this model to demonstrate the effects of IRM-violation on the miss stream, and extrapolate from our results here to the effects of LRU caches on arbitrary distributions.

A. Description of Markov chain

We use a discrete Markov chain, to model the behavior of a tagged file f_* at cache v. We assume the arrival stream is IRM, that $p_{*,v} = \alpha$, and for all other files $p_{i,v} = \beta$, s.t.

$$\beta = \frac{1-\alpha}{N-1}$$

We are interested in the non-IRM traffic characteristics of the miss stream. Following [9], we do so by measuring the *distribution* of the number of requests between two req_*, termed here the *inter-miss distances*.

Our Markov chain consists of states (i, j), where

- $0 \leq i < \infty$. This variable represents the number of cache misses that have occurred for files other than f_*, since f_* entered the cache.
- $j \in \{1, \ldots, |v|$, E(vict), M(iss), A(bsorb)$\}$. This represents the state of f_*. For $i \leq |v|$, f_* is said to be in the i-th location of the cache. Otherwise, the file might be

evicted (state E) or requested after eviction, generating a cache miss (state M). Finally, once a cache miss occurs, we go into the absorbing state (state A).

The transition matrix T can be derived from the transition diagrams, shown in Figure 5. We assume that the system starts in state $(0,1)$, the state of the system after a cache miss for f_*, with f_* at the top of the cache. Each new request at the cache causes a transition in the Markov chain. Denote by T^k the state of the system after k transitions. For each state s, the probability that the system is in state s after k arrivals is expressed by $T^k[(0,1),s]$. Therefore, the probability of an inter-miss distance being h for some $h \in \mathbb{N}$ is

$$P(distance = h) = \sum_{k=1}^{\infty} T^k[(0,1),(h,M)]. \quad (7)$$

For all practical purposes, we need to set a cap for the distances h we are interested in computing, as otherwise the transition matrix is of infinite size. We denote that cap M_{max}.

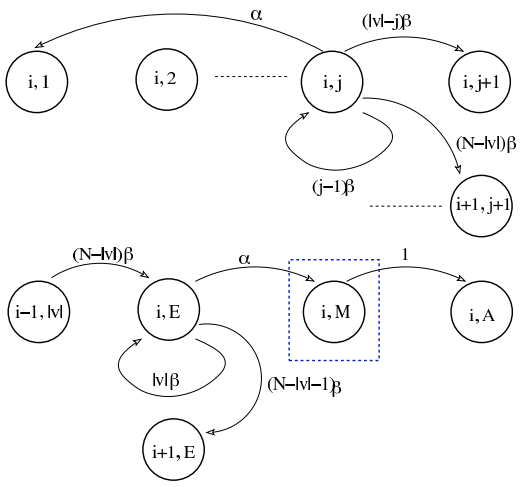

Fig. 5. *Top:* Transition diagram for $1 \leq j \leq |v|$. For the case of $i = M_{max}+1$, because state $(i+1, j+1)$ does not exist we give the transition to $(i, j+1)$ a probability of $(N-j)\beta$. *Bottom:* Transition diagram for $j = E, M, A$. The probability of being in state (i, M) is the probability that the inter-miss distance is i. Because state $(i+1, E)$ does not exist, for the case of $i = M_{max}+1$ we give the transition to (i, E) a probability of $(N-1)\beta$.

B. Characterizing the miss stream using the Markov Model

Using the model described in Section IV-A, we analyzed several scenarios in order to understand how the passing of requests through the cache affected the inter-request distance of the stream. For example, for the case of uniform distribution with $N = 30, |v| = 5, M_{max} = 80$, we compared the inter-request distance distribution of the tagged file, as computed by the Markov model, to the matching distribution for an IRM stream with the same request distribution for file request IDs. The cdf of these distributions is shown in Figure 6.

Let $G^M(x)$ be the inter-request distance distribution cdf for the non-IRM miss stream, as computed by the Markov chain. Furthermore, let $G^I(x)$ be the cdf for an IRM request stream with the same distribution of files requested as the miss stream. As can be seen from Fig. 6, $G^M(x) < G^I(x)$ for all $x < N-1$. More generally, we make the following conjecture:

Conjecture 1: When the incoming request stream at a cache conforms to IRM, $G^M(x) \leq G^I(x)$ for all $x \leq |v|$.

This conjecture is motivated by the observation that the tagged file f_* needs to be evicted from the cache before it can appear again in the miss stream, and so we expect the first $|v|$ misses following a miss for f_* to have a smaller probability of being requests for f_*, compared to an IRM model.

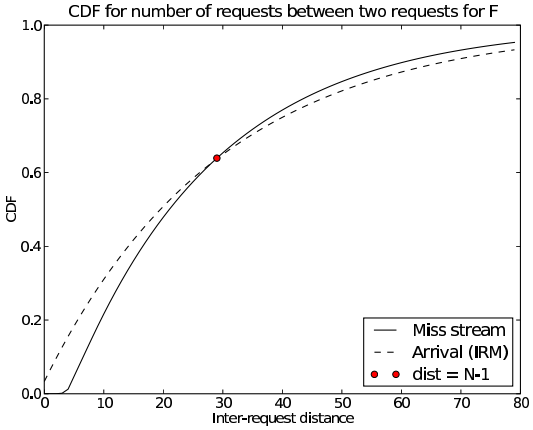

Fig. 6. CDF of inter-arrival distance, for both incoming (IRM) and miss (non-IRM) streams. The distance $N-1$ marks the point after which the negative effects of the cache on the cumulative distribution are no longer felt.

We now proceed to use this Markov model to understand the effects IRM-violation in the miss stream will have on the performance of the next-hop cache. We present the following Theorem, proven in [7]:

Theorem 1: Let f_*, $G^M(k)$ and $G^I(k)$ be defined as above, and assume that each of the remaining files arrives with probability β. Additionally, let $Z^M(k)$ be the the inter-arrival distance distribution (IADD) cdf for all other files[1], and $Z^I(k)$ be the IADD cdf for an IRM request stream with request probability β. Finally, let h_*^M and h_*^I be the hit probability of f_* at cache v for each model. Then, if $Z^M(k) \leq Z^I(k)$ for all $k \leq |v|$, then

$$h_*^M \longrightarrow_{N\to\infty} G^M(|v|-1)$$
$$h_*^I \longrightarrow_{N\to\infty} G^I(|v|-1).$$

Based on Theorem 1 and Conjecture 1, the inter-arrival distance distribution generated by the model can be used to estimate how the hit probability changes for a miss stream which is non-IRM.

We present here our results from testing the case of $N = 50$, $|v| = 3$, for varying values of α, ranging from 0.05 to 0.8, and compare the resulting inter-miss distribution to the IRM equivalent in two ways. First, based on Theorem 1 we estimate

[1]We assume that these files have the same inter-arrival distribution. If not, $Z^M(k)$ takes the maximal value for each k over all the files.

the difference in hit probability between an IRM stream and the miss stream, by computing $G^I(|v|-1) - G^M(|v|-1)$. Second, we took the mean-square difference between the pdfs of both streams, to see how the hit probability is linked to overall distributional difference. The results are shown in Fig. 7.

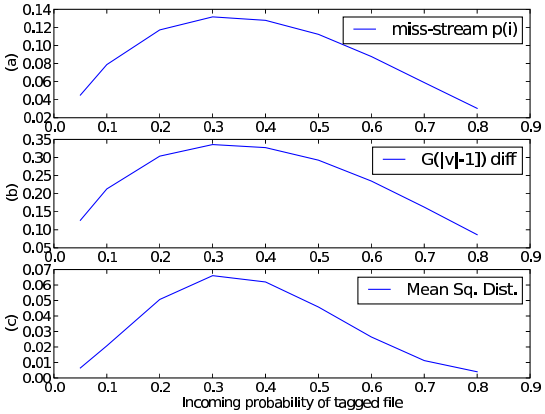

Fig. 7. Evaluating the effects of IRM violation on the inter-request distance distribution, as a function of the incoming probability of the tagged file ($p_* = \alpha$). (a) The fraction of requests for f_* in the miss stream. (b) $G^I(|v|-1) - G^M(|v|-1)$. (c) the mean-square difference between the two PDFs, indicating how different the distributions are.

Figure 7 indicates that there is a clear correlation between, the popularity of req_* in the miss stream, the hit probability difference and the mean-square error of the inter-arrival PDFs. As the popularity of the file in the *miss* stream grows (y axis in Fig. 7.(a)), so too does the difference between IRM and non-IRM distributions in terms of hit probability (Fig. 7.(b)). Also, the changes in the distribution reflect this effect (Fig. 7.(c)).

It is important to note that the error of *a-NET* in predicting the miss probability is proportional to the popularity of the file. Specifically, in the case presented here, we get that $0.0224 \leq \frac{G^M(|v|-1)}{G^I(|v|-1)} \leq 0.0272$. Though popular files are on the high end of this range, the ratios are very similar. This proportionality leads us to conjecture that the prediction error of *a-NET*, in terms of MPR, is mainly one of *scale*, but files retain their relative weight in the request streams at each cache. This conjecture is supported by the results presented in Fig. 1. We saw there that even though the MPR can reach 1.16, the standard deviation in the incoming request streams is approximated with high accuracy by *a-NET*.

Another important conclusion to be derived from these results is that the performance of *a-NET* might differ for different files. Files on the two extremes of the popularity scale in the *arrival* stream will tend to suffer less from the imprecisions of *a-NET*, compared to files in the middle range. It is therefore reasonable to assume that metrics that break down performance based on content attributes, such as popularity, might be more suitable for analyzing MCA generating algorithms than cache-centric metrics, such as MPR. We present one such additional metric next, in Section V.

V. Numerical results

In this section we present additional numerical evaluation of the accuracy of *a-NET*. There are many parameters that affect the accuracy of *a-NET*, and so here we focus on several key parameters: network connectivity, cache size, request distribution and source clustering. In all of our experiments, *a-NET* converged quickly to a fixed point, the exact number of iterations determined by the network diameter and required precision (i.e., the threshold after which changes between iterations were considered negligible). Regarding the latter, we consistently assumed that the total exogenous rate at each node is 10 and halted when the mean-square distance between the results of two iterations was less than 10^{-10}.

Connectivity. In section III-A we saw that an increase in the branch factor improves performance. We tested the effects of increasing the node degree in a cache network on the MPR performance of *a-NET*. We generated random, 400-node graphs with each edge existing in the graph with probability $0.01 \leq p \leq 0.9$, with 500 files distributed according to Zipf distribution with parameter 1. Files were distributed between 10 sources randomly, and sources were randomly placed in each graph. The results are presented in Fig. 8. As expected, the MPR of *a-NET* gets closer to 1 as the connectivity of the graph increases. Note however that this improvement might be a result of the shorter distances between nodes, which limits the aggregation of errors in the prediction of *a-NET*. Further experimentation is required to determine the relative effect of these two factors. Whatever the exact cause, however, increasing connectivity clearly improves the accuracy of *a-NET*.

Source Clustering. Another parameter that might affect performance is the clustering of the sources. When all sources are tightly clustered, the shortest path to each of the sources is the same for the most part from other nodes in the network. This creates a similar behavior to cache trees; specifically, the network contains few *cross streams* - situations where a link or an entire path contains request streams flowing in opposite directions. In our work, we have observed that such occurrences tend to cause *a-NET* to have increased prediction error, and we are currently examining possible explanations for this phenomenon. We demonstrate it here over a 10-by-10 torus with 4 sources, and observed the mean MPR as the distance between sources grew. $|F| = 1000$, $|v| = 50$, files were assigned randomly to sources, and their arrival process was distributed using Zipf with parameter 1. The effects of clustering on the MPR metric are shown in Fig. 9. As can be seen here, there is a slight but gradual increase in MPR as the average distance between sources grows.

Realistic topologies, request distributions, cache size. For evaluation over realistic topologies, we generated transit-stub topologies modeled after the AT&T network, as suggested by Heckmann et al [10], using the GT-ITM tool for topology generation. Here, we compared the prediction of *a-NET* to

the actual system using per-file average number of hops per request. This measure can be considered a system-wide metric, as it takes into account the interactions between caches. For several reasons, it is expected that *a-NET* will predict the number of hops with high accuracy for files on both extremes of the popularity scale, and less so for those in the middle. First, in Section IV we noted that the difference in hit probability is low on both of these extremes. Second, popular requests will mostly require a single hop, while unpopular requests will traverse the shortest path all the way to the source. Such performance can be clearly seen in Figure 10.

We considered the following scenarios, observing the effects of cache size and request distribution:

- $|F| = 500$, $|v| = 50$, distrib. = Zipf with parameter 1.0.
- $|F| = 500$, $|v| = 50$, distrib. = Zipf with parameter 0.6.
- $|F| = 500$, $|v| = 20$, distrib. = Zipf with parameter 1.0.
- $|F| = 250$, $|v| = 50$, distrib. = Zipf with parameter 1.0.

The results are presented in Figure 11. As can be seen here, *a-NET* performs better when the distribution is skewed to a larger degree, i.e. when the Zipf parameter is larger. However, in the examples shown here, the number of files in the system and the cache size seem to have little effect on performance.

For the topologies considered here, the mean error is within range $10 - 15\%$. The diameter of the networks used here was ≤ 10, and so the errors seen here for hop count indicate predicting 1-2 hops less than actually occurs. Further experiments are required to determine how sensitive *a-NET* will be to an increase in network size.

We also observed the arrival distribution at all nodes for these simulations, and calculated the mean square difference of the predicted vs. actual values. As expected, our results show that the mean-square error, averaged over all nodes, is $\leq 10^-6$, regardless of request distribution. This gives strong support to our conjecture that *a-NET* can provide reliable predictions for the incoming request distribution at each node.

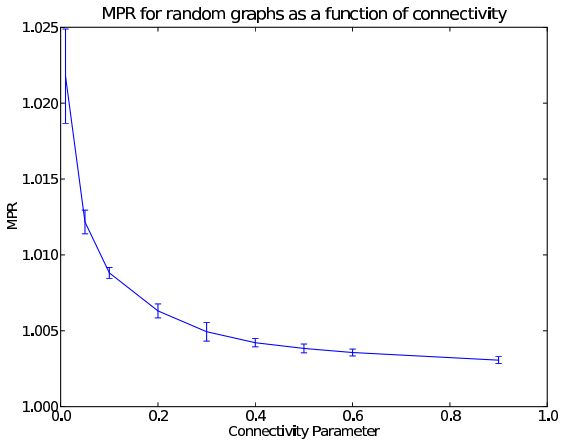

Fig. 8. Mean MPR for random graphs over 400 nodes, as a function of p, the probability that each edge is in the network. The mean is taken over 10 simulations for each p, with 95% confidence intervals showing.

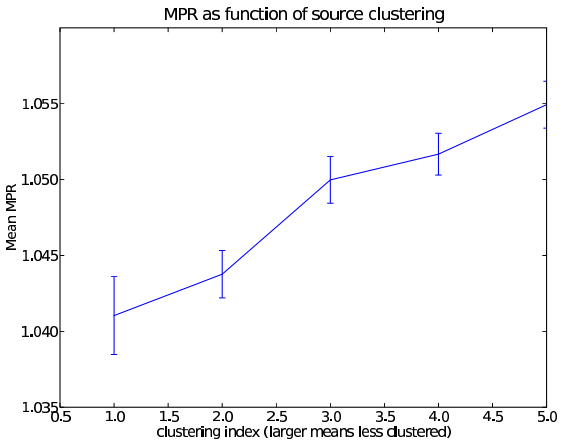

Fig. 9. Mean MPR for 10-by-10 torus networks, as a function of the source clustering. There were 4 sources positioned at the corners of a square, and the x-axis specifies the length of the square side in terms of no. of hops. The mean is taken over 6 simulations for each clustering, with 95% confidence intervals showing.

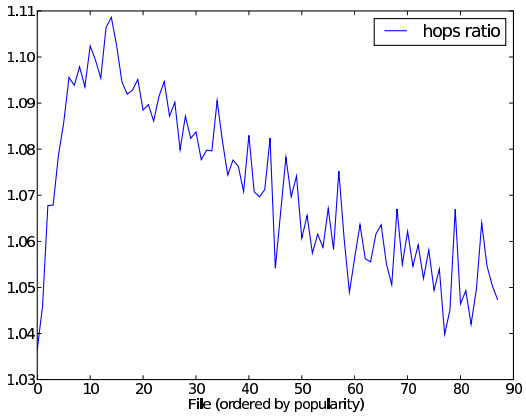

Fig. 10. Ratio of no. of hops for files stored at a given source (Sim/Approx). Files are ordered by popularity. As can be seen, highly-popular and highly-unpopular files are approximated with greater accuracy.

VI. Related Work

The approach we pursued here draws much of its inspiration from Kelly's well-known reduced-load approximation for computing blocking probabilities in circuit-switched networks [11]. The reduced load approximation computes this blocking probability by assuming, similar to our approach here, that the call arrival process to a link *j* behaves as an *independent* process. This allows the network to be analyzed as a set of independent links, coupled only through the blocking rates at each link.

Exogenous request streams were assumed throughout the paper to conform to IRM. However, there are alternative models for request patterns at single caches. Panagakis et. al. [12] present approximate analysis for streams that have short term correlations for requests. In their model, the arrival

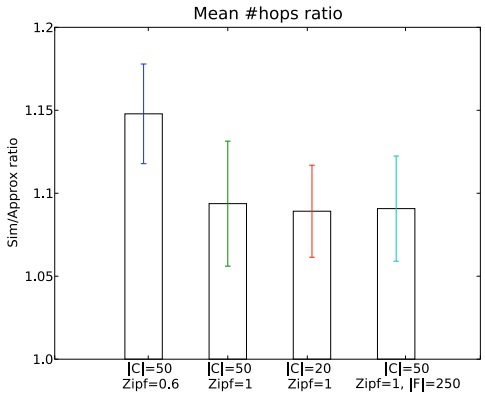

Fig. 11. Mean ratio of no. of hops per request. $|F| = 500$ unless specified otherwise, and $|C|$ is the cache size. Results were obtained by random placement of 4 sources and random association of files with sources, over 6 simulation. Error bars represent 95% confidence intervals.

process is IRM, with the exception that the k most recent requests have a higher probability of arriving next at the cache than other requests. The justification for this model is that such correlations have been found in web-traces. However, as we have seen, cache networks experience the opposite effect, with recent requests *less* likely to arrive next.

Another alternative is Stack Depth Distribution (SDD) (for example, see [13]). With this model, the stream of requests is characterized as a distribution $\vec{h} = (h_i)_{i=1}^{\infty}$ over the cache slots in a cache of infinite capacity, where h_i is the probability that the next cache hit will be at slot i. In this model, all information regarding the individual files being requested is ignored or unavailable.

For IRM traffic, there are additional algorithms of equal complexity to *a-LRU* that compute the hit probability at a single cache, but these are not as easily used or as informative. For example, Flajolet et. al. [14] presented an integral solution for the cache approximation problem, which can be solved numerically to produce the hit ratio. However, there is no straight-forward manner by which to observe the behavior of each file with this approach.

Che et. al. developed a model for a two-level LRU-based cache hierarchy [5], by using a "mean field" approximation of each cache. They associate each file in each cache with a *constant* time, representing "the maximum inter-arrival time between two adjacent requests for [the] document without a cache miss". They justify this model by claiming that as the number of files in the system goes to infinity, this assumption becomes reasonable. This technique was subsequently leveraged in [6] to analyze cache coordination policies for cache hierarchies. Neither paper provides much simulation support for this model, and the approach is limited to 2-level cache hierarchies, and cannot be easily extended to larger tree sizes.

VII. Conclusions and Future Research

In this paper we presented *a-NET*, our MCA generating algorithm, designed to evaluate performance measures such as miss probability per cache and system-wide hops per request. In general, *a-NET* under-estimates the miss probability of caches, for which we have presented analytical support. The accuracy of our approximation is affected primarily by IRM-violations in the cache miss streams, making the approximation most accurate with cache trees with large branch factors and, in general, with highly connected topologies. We have also seen that *a-NET* is highly accurate in predicting the incoming distribution at all nodes, even when the MPR is large.

a-NET is designed to approximate the behavior of a cache network with static routing tables, while a more realistic model would consider dynamic routing as well. Adapting *a-NET* to such a model is a challenging task we plan to address next. *a-NET* was presented here for LRU caches, but can be used with any SCA algorithm that receives the steady state distribution of request arrivals, and returns the probability of each file to be in the cache. The Markov model presented in section IV can also be adapted, with minimal changes, to other replacement policies, such as FIFO. We are currently working on several ways in which to leverage these properties to expand our understanding of cache networks.

References

[1] B. Ahlgren, M. DAmbrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," in *ReArch'08 Workshop*. ACM, 2008.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*. ACM, 2009, pp. 1–12.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM*, 2007.

[4] A. Dan and D. F. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *SIGMETRICS*, 1990, pp. 143–152.

[5] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *IEEE INFOCOM*, 2001, pp. 1416–1424.

[6] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, pp. 609–634, 2006.

[7] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," UMass Amherst, MA, Tech. Rep. UM-CS-2009-037, 2009.

[8] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM*, 1999, pp. 126–134.

[9] R. Fonseca, V. Almeida, M. Crovella, and B. Abrahao, "On the intrinsic locality properties of web reference streams," in *IEEE INFOCOM*, 2003.

[10] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On realistic network topologies for simulation," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 2003, pp. 28–32.

[11] F. Kelly, "Loss networks," *The Annals of Applied Probability*, vol. 1, no. 3, pp. 319 – 378, 1991.

[12] A. Panagakis, A. Vaios, and I. Stavrakakis, "Approximate analysis of LRU in the case of short term correlations," *Comput. Netw.*, vol. 52, no. 6, pp. 1142–1152, 2008.

[13] H. Levy and R. J. T. Morris, "Exact analysis of Bernoulli superposition of streams into a least recently used cache," *IEEE Trans. Softw. Eng.*, vol. 21, no. 8, pp. 682–688, 1995.

[14] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Appl. Math.*, vol. 39, no. 3, pp. 207–229, 1992.