Distributed Selfish Caching

Nikolaos Laoutaris, Georgios Smaragdakis, Azer Bestavros,
Ibrahim Matta, *Senior Member*, *IEEE*, and Ioannis Stavrakakis, *Fellow*, *IEEE*

Abstract—Although cooperation generally increases the amount of resources available to a community of nodes, thus improving individual and collective performance, it also allows for the appearance of potential mistreatment problems through the exposition of one node's resources to others. We study such concerns by considering a group of independent, rational, self-aware nodes that cooperate using online caching algorithms, where the exposed resource is the storage at each node. Motivated by *content networking* applications—including Web caching, content delivery networks (CDNs), and peer-to-peer (P2P)—this paper extends our previous work on the offline version of the problem, which was conducted under a game-theoretic framework and limited to object replication. We identify and investigate two causes of mistreatment: 1) cache *state interactions* (due to the cooperative servicing of requests) and 2) the adoption of a *common scheme* for cache management policies. Using analytic models, numerical solutions of these models, and simulation experiments, we show that online cooperation schemes using caching are fairly robust to mistreatment caused by state interactions. To appear in a substantial manner, the interaction through the exchange of miss streams has to be very intense, making it feasible for the mistreated nodes to detect and react to exploitation. This robustness ceases to exist when nodes fetch and store objects in response to remote requests, that is, when they operate as Level-2 caches (or proxies) for other nodes. Regarding mistreatment due to a common scheme, we show that this can easily take place when the "outlier" characteristics of some of the nodes get overlooked. This finding underscores the importance of allowing cooperative caching nodes the flexibility of choosing from a diverse set of schemes to fit the peculiarities of individual nodes. To that end, we outline an emulation-based framework for the development of mistreatment-resilient distributed selfish caching schemes.

Index Terms—Selfish caching, cooperative caching, overlay networks, peer-to-peer networks, content distribution networks.

1 Introduction

1.1 Background, Motivation, and Scope

Network applications often rely on distributed resources available within a cooperative grouping of nodes to ensure scalability and efficiency. Traditionally, such groupings are dictated by an overarching common strategic goal. For example, nodes in a content delivery network (CDN) such as Akamai or Speedera cooperate to optimize the performance of the overall network, whereas IGP routers in an Autonomous System (AS) cooperate to optimize routing within the AS.

More recently, however, new classes of network applications have emerged for which the grouping of nodes is more "ad hoc" in the sense that it is not dictated by organizational boundaries or strategic goals. Examples include overlay protocols [3], [6] and peer-to-peer (P2P) applications. Two distinctive features of such applications are that 1) individual nodes are autonomous, and as such, their membership in a group is motivated solely by the selfish goal of *benefiting* from that group, and 2) group membership is warranted only as long as a node is interested in being part of the application or protocol, and as such, group membership is expected to be fluid. In light of these characteristics, an important question is this: *Are protocols and applications that rely on sharing of distributed resources appropriate for this new breed of ad hoc node associations?*

In this paper, we answer this question for networking applications, whereby the distributed resource being shared among a group of nodes is *storage*. Although our work and methodology are applicable for a wide range of applications that rely on distributed shared storage, we target the distribution of voluminous content as our application of choice.[1] In particular, we consider a group of nodes that store information objects and make them available to their local users, as well as to remote nodes. A user's request is first received by the local node. If the requested object is stored locally, it is returned to the requesting user immediately, thereby incurring a minimal access cost. Otherwise, the requested object is searched for and fetched from other nodes of the group at a potentially higher access cost. If the object cannot be located anywhere in the group, it is retrieved from an origin server, which is assumed to be outside the group, thus incurring a maximal access cost.

Under an *object replication* model, once selected for replication at a node, an object is stored permanently at that node (that is, the object cannot be replaced later). In [18], we established the vulnerability of *socially optimal* (SO) object replication schemes in the literature to *mistreatment* problems. We define a mistreated node to be a node whose

- *N. Laoutaris is with Harvard University, Maxwell Dorkin 207, 33 Oxford Street, Cambridge, MA 02138. E-mail: nlaout@eecs.harvard.edu.*
- *G. Smaragdakis, A. Bestavros, and I. Matta are with the Computer Science Department, Boston University, 111 Cummington Street, Boston, MA 02215. E-mail: {gsmaragd, best, matta}@cs.bu.edu.*
- *I. Stavrakis is with the Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilisia 157-84, Athens, Greece. E-mail: istavrak@di.uoa.gr.*

Manuscript received 21 Feb. 2006; revised 20 Sept. 2006; accepted 20 Sept. 2006; published online 8 Feb. 2007.
Recommended for acceptance by S. Das.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0037-0206.
Digital Object Identifier no. 10.1109/TPDS.2007.1076.

[1] Unlike content distribution for static (typically small) Web objects such as HTML Web pages and images, voluminous content requires treating the storage as a limited resource [19].

1045-9219/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society

access cost under SO replication is higher than the minimal access cost that the node can guarantee under greedy local (GL) replication. Unlike centrally designed/controlled groups where all constituent nodes have to abide by the ultimate goal of optimizing the social utility of the group, an autonomous selfish node will not tolerate such a mistreatment. Indeed, the emergence of such mistreatments may cause selfish nodes to secede from the replication group, resulting in severe inefficiencies for both the individual users, as well as the entire group.

In [18], we resolved this dilemma by proposing a family of *equilibrium* (EQ) object placement strategies that 1) avoid the mistreatment problems of SO, 2) outperform GL by claiming available "cooperation gain" that the GL algorithm fails to utilize, and 3) are implementable in a distributed manner, requiring the exchange of only a limited amount of information. The EQ strategies were obtained by formulating the *Distributed Selfish Replication* (DSR) game and devising a distributed algorithm that is always capable of finding pure Nash EQ strategies for this particular game. Apart from this work, we are aware of only two additional works on game-theoretic aspects of replication, one is by Chun et al. [4] (DSR under infinite storage capacities), and the other is by Erçetin and Tassiulas [9] (market-based resource allocation in content delivery); we are not aware of any previous work on distributed selfish caching.

1.2 Distributed Selfish Caching

Proactive replication strategies are not practical in a highly dynamic content networking setting, which is likely to be the case for most of the Internet overlays and P2P applications we envision. This is due to a variety of reasons:

1. Fluid group membership makes it impractical for nodes to decide what to replicate based on which (and where) objects are replicated in the group.
2. Access patterns, as well as access costs, may be highly dynamic (due to bursty network/server loads), necessitating that the selection of replicas and their placement be done continuously, which is not practical.
3. Both the identification of the appropriate reinvocation times [22] and the estimation of the nonstationary demands (or, equivalently, the timescale for a stationarity assumption to hold) [12] are nontrivial problems.
4. Content objects may be dynamic and/or may expire, necessitating the use of "pull" (that is, on-demand *caching*) as opposed to "push" (that is, proactive replication) approaches.

Using on-demand caching is the most widely acceptable and natural solution to all of these issues because it requires no a priori knowledge of local/group demand patterns and, as a consequence, responds dynamically to changes in these patterns over time (for example, the introduction of new objects, reduction in the popularity of older ones, etc.).

Therefore, in this paper, we consider the problem of *Distributed Selfish Caching* (DSC), which can be seen as the *online* counterpart to the DSR problem. In DSC, we adopt an *object caching* model, whereby a node employs a demand-driven temporary storage of objects, combined with replacement. At this juncture, it is important to note that we make a clear distinction between *replication* and *caching*. Although these terms may be seen as similar (and indeed used interchangeably in much of the literature), we note that for our purposes they carry quite different meanings and implications. Replication amounts to maintaining permanent copies, whereas caching amounts to maintaining temporary copies. This changes fundamentally the character and the methodologies used in analyzing DSR and DSC.

1.3 Causes of Mistreatments under DSC

We begin our examination of DSC by considering the operational characteristics of a group of nodes involved in a distributed caching solution. This examination will enable us to identify two key culprits for the emergence of mistreatment phenomena.

First, we identify the mutual *state interaction* between replacement algorithms running on different nodes as the prime culprit for the appearance of mistreatment phenomena. This interaction takes place through the so-called "remote hits." Consider nodes v and u and object o. A request for object o issued by a user of v that cannot be served at v but could be served at u is said to have incurred a *local miss* at v but a *remote hit* at u. Consider now the implications of the remote hit at u. If u does not discriminate between hits due to local requests and hits due to remote requests, which is the default behavior of the Internet Cache Protocol (ICP)/Squid Web cache [8] and other systems (for example, Akamai Content Distribution Network, IBM Olympic Server Architecture), then the remote hit for object o will affect the state of the replacement algorithm in effect at u. If u is employing Least Recently Used (LRU) replacement, then o will be brought to the top of the LRU list. If it employs Least Frequently Used (LFU) replacement, then its frequency will be increased, and so on with other replacement algorithms [26]. If the frequency of remote hits is sufficiently high, for example, because v has a much higher local request rate and, thus, sends an intense miss stream to u, then there could be performance implications for the second: u's cache may get invaded by objects that follow v's demand, thereby depriving the user's of u of valuable storage space for caching their own objects. This can lead to the mistreatment of u, whose cache is effectively "hijacked" by v.

Moving on, we identify a second, less-anticipated culprit for the emergence of mistreatment in DSC. We call it the *common scheme* problem. To understand it, one has to observe that most of the work on cooperative caching has hinged on the fundamental assumption that all nodes in a cooperating group adopt a common scheme. We use the word "scheme" to refer to the combination of 1) the employed *replacement algorithm*, 2) the employed *request redirection algorithm*, and 3) the employed *object admission algorithm*. Cases 1 and 2 are more or less self-explanatory. Case 3 refers to the decision of whether to cache locally an incoming object after a local miss. The problem here is that the adoption of a common scheme can be beneficial to some of the nodes of a group but harmful to others, particularly to nodes that have special characteristics that make them "outliers." A simple case of an outlier is a node that is situated further away from the center of the group, where most nodes lie. Here, distance may have a topological/affine meaning (for example, number of hops or propagation delay), or it may relate to dynamic performance characteristics (for example, variable throughput or latencies due to load conditions on network links or server nodes). Such an outlier node cannot rely on the other nodes for fetching objects at a small access cost and, thus, prefers to keep local copies of all incoming objects. The rest of the nodes, however, as long as they are close enough to each other, prefer not to cache local copies of incoming objects that already exist elsewhere in the group. Since such objects can be fetched from remote nodes at a small access cost, it is better to preserve the local storage for keeping objects that do not exist in the group and, thus, must be fetched from the origin server at a high access cost. In this setting, a common scheme is bound to mistreat either the outlier node or the rest of the group.

2 Problem Setting and Summary of Results

In this section, we first introduce the setting in which we study DSC and then summarize our results.

2.1 Definitions and Notation

Let o_i, $1 \leq i \leq N$, and v_j, $1 \leq j \leq n$, denote the ith unit-sized object and the jth node, and let $O = \{o_1, \dots, o_N\}$ and $V = \{v_1, \dots, v_n\}$ denote the corresponding sets. Node v_j is assumed to have a storage capacity of up to C_j unit-sized objects, a total request rate of λ_j (total number of requests per unit time, across all objects), and a demand described by a probability distribution over O, $\vec{p}_j = \{p_{1j}, \dots, p_{Nj}\}$, where p_{ij} denotes the probability of object o_i being requested by the local users of node v_j. Successive requests are assumed to be independent and identically distributed.[2] Later in this paper, we make the specific assumption that the popularity of objects follows a power-law profile, that is, the ith most popular object is requested with probability $p_i = K/i^a$. Such popularity distributions occur in many measured workloads [2], [23] and, although used occasionally in our work (for example, in Section 3.1 to simplify an analytic argument and in Section 4 for producing numerical results), they do not constitute a basic assumption in the sense that mistreatment can very well occur with other demand distributions that do not follow such a profile.

Let t_l, t_r, and t_s denote the access cost paid for fetching an object locally, remotely, or from the origin server, respectively, where $t_s > t_r > t_l$.[3] User requests are serviced by the closest node that stores the requested object along the following chain: local node, group, and origin server. Each node employs a replacement algorithm for managing the content of its cache and employs an object admission algorithm for accepting (or not) incoming objects.

2.2 Summary of Results

In addition to defining the DSC problem and the causes of mistreatment, this paper presents a number of concrete results regarding each one of these causes. These results are intended to be used as basic design guidelines on dealing with selfishness in current and future caching applications.

Mistreatment Due to Cache State Interaction. Regarding the state interaction problem, our investigations answer the following basic question: "*Could and under which schemes do mistreatments arise in a DSC group?*" More specifically:

- We show that state interactions occur when nodes do not discriminate between local and remote *hits* upon updating the state of their replacement algorithms.
- To materialize, state interactions require a substantial request rate imbalance, that is, one or more "overactive" nodes must generate disproportionally more requests than the other nodes. Even in this case, mistreatment of less active nodes depends on the amount of storage that they posses: mistreatment occurs when these nodes have abundant storage; otherwise, they are generally immune to or even benefit from the existence of overactive nodes.
- Comparing caching and replication with regard to their relative sensitivities to request rate imbalance, we show that caching is much more robust than replication.
- Regarding the vulnerability of different replacement algorithms, we show that "noisier" replacement algorithms are more prone to state interactions. In that regard, we show that LRU is more vulnerable than LFU.
- Even the most vulnerable LRU replacement is quite robust to mistreatment as it requires a very intense miss stream in order to force a mistreated node to maintain locally unpopular objects in its cache (thus depriving it of cache space for locally popular objects). In particular, the miss stream has to be strong enough to counter the sharp decline in the popularity of objects in typically skewed workloads.
- Robustness to mistreatment due to state interaction evaporates when a node operates as a Level-2 (L2) cache [31] for other nodes. L2 caching allows all remote requests (whether they hit or miss) to affect the local state (as opposed to only hits under non-L2 caching), leading to a vulnerability level that approaches the one under replication.

Mistreatment Due to the Use of a Common Scheme. We classify cooperative caching schemes into two groups: *Single Copy* (SC) schemes, that is, schemes where there can be at most one copy of each distinct object in the group (two examples of SC schemes are HASH-based caching [28] and LRU-SC [10]), and *Multiple Copy* (MC) schemes, that is, schemes where there can be multiple copies of the same object at different nodes.

- We show that the relative performance ranking of SC and MC schemes changes with the "tightness" of a cooperative group. SC schemes perform best when the internode distances are small compared to the

[2] The Independent Reference Model (IRM) [5] is commonly used to characterize cache access patterns [1], [2]. The impact of temporal correlations was shown in [12] and [27] to be minuscule, especially under typical Zipf-like object popularity profiles.

[3] The assumption that the access cost is the same across all node pairs in the group is made only for the sake of simplifying the presentation. Our results can be adapted easily to accommodate arbitrary internode distances.

distance to the origin server; in such cases, the maintenance of multiple copies of the same object becomes unnecessary.[4] MC schemes improve progressively as the internode distances increase and eventually outperform the SC schemes.

- We demonstrate a case of mistreatment due to a common scheme by considering a tight group of nodes that operate under an SC scheme and a unique outlier node that has a larger distance to the group. We show that this node is mistreated if it is forced to follow the same SC scheme.

Toward Mistreatment-Resilient DSC Schemes. More constructively, we present a framework for the design of mistreatment-resilient DSC schemes. Our framework allows individual nodes to decide autonomously (that is, without having to trust any other node or service) whether they should stick to or secede from a DSC caching group based on whether or not their participation is beneficial to their performance compared to a selfish greedy scheme. Resilience to mistreatments is achieved by allowing a node to emulate the performance gain possible by switching from one scheme to another or by adapting some control parameters of its currently deployed DSC scheme. We use a simple control-theoretic approach to dynamically parameterize the DSC scheme in use by a local node. We evaluate the performance of our solution by considering caching in wireless mobile nodes [32] where distances and download rates depend on mobility patterns. We show that our adaptive schemes can yield substantial performance benefits, especially under skewed demand profiles.

3 Mistreatment Due to State Interaction: Analysis

Our goal in this section is to understand the conditions under which mistreatment may arise as a result of (cache) state interactions. We start in Section 3.1 with a replacement-agnostic model that focuses on the rate imbalance (between the local request stream and the remote miss stream) necessary for mistreatment to set in. Next, in Section 3.2, we present a more detailed analytical model that allows for the derivation of the average access cost in a distributed caching group composed of n nodes that operate under LRU replacement.

3.1 General Conditions

We would like to determine the level of *request rate imbalance* that is necessary for mistreatment to be feasible. We model this imbalance through the ratio λ_n/λ_j, where λ_j denotes the request rate of any normally behaving node v_j, and λ_n denotes the request rate of an overactive node, which we use to instigate mistreatment problems. As a convention, we assume this overactive node to be the last (nth) node of the group.

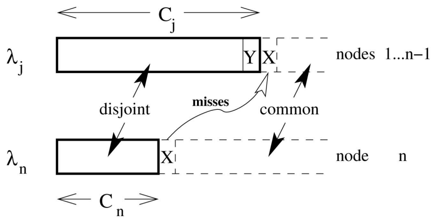


Fig. 1. Reference model for the study of mistreatment due to state interaction.

We focus on the interaction between v_j and v_n. Fig. 1 shows a particular choice of demand patterns that fosters the occurrence of mistreatment. The initial most popular objects in $\vec{p}_j$ and $\vec{p}_n$ up to the two capacities (C_j for v_j and C_n for v_n) are completely disjoint, whereas the remaining ones in the middle part of the two distributions are identical; both demands are assumed to be a power law with parameter a. Let X denote the most popular object that is common to both distributions. A boundary condition for the occurrence of mistreatment can be obtained by considering the ratio λ_n/λ_j that results in a switch of ranking between X and Y at v_j, where Y denotes the least most popular object that would be kept in the cache of v_j under a perfect ranking of objects according to the local demand if no miss stream was received. To derive the condition for the switch, we first note that X is the $(C_j+1)\mathrm{th}$ most popular object for v_j and the $(C_n+1)\mathrm{th}$ most popular one for v_n. Y is the $C_j\mathrm{th}$ most popular object for v_j. Let $f(n)$ denote a function that captures the operation of different object location mechanisms in a group of n nodes (used for locating and retrieving objects from remote nodes). For example, $f(n)=1$ can be used for modeling request flooding (following a local miss, a request is sent to all other nodes in the group); $f(n)=1/(n-1)$ can be used for modeling index-based mechanisms [10] (following a local miss, a request is sent to only one of the nodes that appear to be storing the object according to some index). The boundary condition for the occurrence of the switch can be written as follows:

$$\begin{gathered}\lambda_j p_{C_j} \le \lambda_j p_{C_j+1} + \lambda_n p_{C_n+1} f(n) \Rightarrow \\ \lambda_j \frac{1}{(C_j)^a} \le \lambda_j \frac{1}{(C_j+1)^a} + \lambda_n \frac{1}{(C_n+1)^a} f(n) \Rightarrow \\ \frac{\lambda_n}{\lambda_j} \ge \frac{(C_n+1)^a}{f(n)} \left(\frac{1}{C_j^a} - \frac{1}{(C_j+1)^a} \right).\end{gathered} \tag{1}$$

Writing a continuous approximation for the rate of change of $1/C^a$ with respect to C, we get

$$\begin{gathered}\frac{d\left(\frac{1}{C^a}\right)}{dC} = \frac{\frac{1}{C^a} - \frac{1}{(C+1)^a}}{C-(C+1)} \approx -a \cdot \frac{1}{C^{1+a}} \Rightarrow \\ \frac{1}{C^a} - \frac{1}{(C+1)^a} \approx a \cdot \frac{1}{C^{1+a}}.\end{gathered} \tag{2}$$

Using the approximation from (2) on (1), we obtain

$$\frac{\lambda_n}{\lambda_j} \ge \frac{(C_n+1)^a}{f(n)} \cdot a \frac{1}{(C_j)^{1+a}} \sim \frac{a}{f(n) C_j} \left(\frac{C_n}{C_j} \right)^a. \tag{3}$$

Equation (3) states that the amount of imbalance in request rates $(\frac{\lambda_n}{\lambda_j})$ that is required for the occurrence of mistreatment is 1) increasing with C_n, 2) decreasing with C_j, and 3) increasing when request flooding is employed for locating remote objects (in this case, all the nodes get the full miss stream from v_n; otherwise, the miss stream weakens by being split into $n-1$ parts).

Now, assume that as a result of the received miss stream, k objects of v_j are switched (objects with IDs $C_j, \dots, C_j-k+1$ evicted, objects $C_j+1, \dots, C_j+k$ inserted); k can be

[4] We do not consider load balancing concerns in this study.

computed from a condition similar to that in (1). Define the $Loss$ of v_j as the reduction in the probability mass of the objects that it caches locally:

$$\begin{aligned} Loss = & \sum_{i=C_j-k+1}^{C_j} p_i - \sum_{i=C_j+1}^{C_j+k} p_i = \\ & K \cdot \left(H_{C_j}^{(a)} - H_{C_j-k}^{(a)} - H_{C_j+k}^{(a)} + H_{C_j}^{(a)} \right) = \\ & K \cdot \left(2H_{C_j}^{(a)} - H_{C_j-k}^{(a)} - H_{C_j+k}^{(a)} \right), \end{aligned} \tag{4}$$

where K is the normalization constant of the power-law distribution $p_i = K/i^a$. The generalized harmonic number $H_C^{(a)}$ can be approximated by its integral expression (see [30]) $H_C^{(a)} = \sum_{i=1}^{C} \frac{1}{i^a} \approx \int_1^C 1/l^a dl = \frac{C^{1-a}-1}{1-a}$. Plugging this into (4), we obtain

$$Loss = K\left(2\frac{{C_j}^{1-a}-1}{1-a} - \frac{(C_j-k)^{1-a}-1}{1-a} - \frac{(C_j+k)^{1-a}-1}{1-a} \right). \tag{5}$$

From (5), it is clear that as C_j increases, both $C_j - k$ and $C_j + k \to C_j$, thus leading to $Loss \to 0$. Combining our observations from (3) and (5), we conclude that the *occurrence* of mistreatment is fostered by small C_n and large C_j. Its magnitude, however, decreases with C_j. Therefore, practically, it is in the intermediate values of C_j that mistreatment can arise in a substantial manner.

3.2 Analysis of Mistreatment under LRU Replacement

In the remainder of this section, our objective will be to derive the steady-state hit probabilities $\vec{\pi}_j = \{\pi_{1j}, \ldots, \pi_{Nj}\}$, where π_{ij} denotes the steady-state probability of finding object o_i at node v_j that operates under LRU replacement. We will then use these results for studying mistreatment in the context of LRU.

Let $\vec{\pi} = LRU(\vec{p}, C)$ denote a function that computes the steady-state object hit probabilities for a single LRU cache in isolation, given the cache size and the demand distribution. Due to the combinatorial hardness of analyzing LRU replacement, it is difficult to derive an exact value for $\vec{\pi}$; there are, however, several methods for computing approximate values for it (see, for example, [15] and references therein). In this paper, we employ the approximate method of Dan and Towsley in [7] that provides an accurate estimation of $\vec{\pi}$ through an iterative algorithm that incurs $O(NC)$ time complexity. Having computed $\vec{\pi}_j$, $\forall v_j \in V$, we can obtain the per-node access cost $cost_j$, as well as the social cost of the entire group $cost_{soc} = \sum_{\forall v_j} cost_j$, by using

$$\begin{aligned} & cost_j = \\ & \sum_{i=1}^{N} p_{ij} \cdot \left[\pi_{ij} \cdot t_l + (1-\pi_{ij}) \cdot \pi_{i-j} \cdot t_r + (1-\pi_{ij}) \cdot (1-\pi_{i-j}) \cdot t_s \right], \\ & \quad \text{where} \quad \pi_{i-j} = 1 - \prod_{\forall j' \neq j} (1-\pi_{ij'}). \end{aligned} \tag{6}$$

In this equation, π_{i-j} denotes the probability of finding o_i in any node of the group other than v_j.

We can obtain $\vec{\pi}_j$ by using the $LRU(\cdot,\cdot)$ function for isolated caches as our basic building block and taking into consideration the impact on the local state of the hits caused by remote requests. Deriving an exact expression for these added hits based on the involved cache states is intractable, as it leads to state-space explosion. Therefore, we turn to approximate techniques and, in particular, to techniques that consider the expected values of the involved random variables instead of their exact distributions. The basic idea of our approach is to capture these added hits by properly modifying the input to the $LRU(\cdot,\cdot)$ function.

Remote hits can be considered simply as an additional request that augment the local demand, thereby creating a new aggregate demand for the $LRU(\cdot,\cdot)$ function, as explained later. The idea of modifying the input of a simpler system to capture a policy aspect of a more complex system and then using the modified simpler system to study the more complex one has been employed frequently in the past [14]. Since the remote hits are shaped by the cache states, which are coupled due to the exchanges of miss streams, an iterative procedure is followed for the derivation of the per-node steady-state vectors and access costs. The uncoupled solution (corresponding to nodes operating in isolation) is obtained first and is refined progressively by taking into account the derived states and the cooperative servicing of the misses. The resulting approximate analytical model for predicting the average access cost in a distributed caching group is described below. In the next section, we show that the results produced from this model match quite well with the results obtained through simulations.

The iterative procedure is given as follows:

1. For each node v_j, compute $\vec{\pi}_j^{(0)} = LRU(\vec{p}_j, C_j)$, that is, assume no state interaction among the different nodes.
2. Initiate iteration. At the kth iteration, the aggregate demand distribution for v_j, $\vec{p_j}^{(k)} = \{p_{ij}^{(k)}\}$, $1 \le i \le N$, is given by

$$p_{ij}^{(k)} = \frac{\lambda_j \cdot p_{ij} + \sum_{j'=1, j' \neq j}^{n} \lambda_{j'} \cdot p_{ij'} \cdot (1-\pi_{ij'}^{(k-1)}) \cdot \left[\frac{\left(\pi_{ij}^{(k-1)}\right)^2}{\sum_{j''=1, j'' \neq j'}^{n} \pi_{ij''}^{(k-1)}} \right]_{\pi_{ij}^{(k-1)}}^{+}}{\sum_{i'=1}^{N} \left(\lambda_j \cdot p_{i'j} + \sum_{j'=1, j' \neq j}^{n} \lambda_{j'} \cdot p_{i'j'} \cdot (1-\pi_{i'j'}^{(k-1)}) \cdot \left[\frac{\left(\pi_{i'j}^{(k-1)}\right)^2}{\sum_{j''=1, j'' \neq j'}^{n} \pi_{i'j''}^{(k-1)}} \right]_{\pi_{i'j}^{(k-1)}}^{+} \right)} \tag{7}$$

The function $[x]_y^+$ returns 0 if $y = 0$ and x otherwise.[5] The steady-state vector of object hit probabilities for v_j at iteration k can be obtained from $\vec{\pi}_j^{(k)} = LRU(\vec{p_j}^{(k)}, C_j)$.

[5] This function is used to ensure correctness when the denominator $\sum_{j''=1, j'' \neq j'}^{n} \pi_{ij''}^{(k-1)}$ becomes zero. Notice that the nominator $\pi_{ij}^{(k-1)}$ is included in the denominator, so when $\pi_{ij}^{(k-1)} > 0$, the denominator is guaranteed to be nonzero.

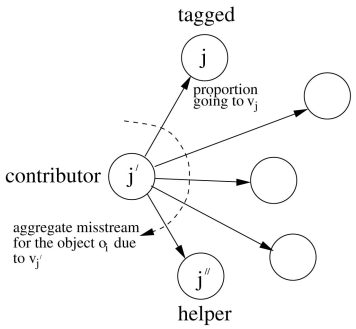

tagged
j
proportion going to v$_j$
contributor
j$'$
aggregate misstream for the object o$_i$ due to v$_{j'}$
j$''$
helper

Fig. 2. Graphical illustration for the explanation of (7).

3. Perform convergence test. If $|\vec{\pi}_j^{(k)} - \vec{\pi}_j^{(k-1)}| < \vec{\epsilon}$ for all v_j, $1 \le j \le n$, then set $\vec{\pi}_j = \vec{\pi}_j^{(k)}$, and compute the per-node access costs from (6); $\vec{\epsilon}$ denotes a user-defined tolerance for the convergence of the iterative method. Otherwise, set $\vec{\pi}_j^{(k-1)} = \vec{\pi}_j^{(k)}$ and $\vec{p}_j^{(k-1)} = \vec{p}_j^{(k)}$, and perform another iteration by returning to Step 2.

The nominator of (7) adds the requests generated by the local population of v_j for object o_i to the requests for the same object due to the $n-1$ miss streams from all other nodes that create hits at v_j. The explanation of the circumstances under which such hits exist goes as follows (see also Fig. 2): A request for o_i received at the *contributor* node $v_{j'}$ (probability $p_{ij'}$) affects the *tagged* node v_j if the request cannot be serviced at the contributor node (probability $(1 - \pi_{ij'}^{(k-1)})$), can be serviced at the tagged node (probability $\pi_{ij}^{(k-1)}$), and is indeed serviced by the tagged node and not by any other *helper* node $v_{j''}$ that can potentially service it (probability $\pi_{ij}^{(k-1)} / \sum_{j''=1, j''\neq j'}^{n} \pi_{ij''}^{(k-1)}$, that is, the model assumes that when more than one helper nodes can offer service, the request is assigned uniformly to any one of them).

Before we conclude this section, we note that our aforementioned analysis could be construed as providing a lower bound of the intensity of mistreatment assuming that the proxy is configured such that only *one* peer (proxy cache) replies to a remote request. Mistreatment could be more severe if, upon a local miss, requests are routed to more than one proxy, which is the case in many real systems [8].

4 Mistreatment Due to State Interaction: Evaluation

In this section, we use a combination of simulation experiments and numerical solutions of the analytical model developed in the previous section to explore the design space of distributed caching with respect to its vulnerability to the onset of mistreatment as a result of the state interaction phenomenon. We start by validating the accuracy of the analytical model in Section 3.2 and follow that with an examination of various dimensions of the design space for distributed caching, including a comparative evaluation of mistreatment in caching versus replication.

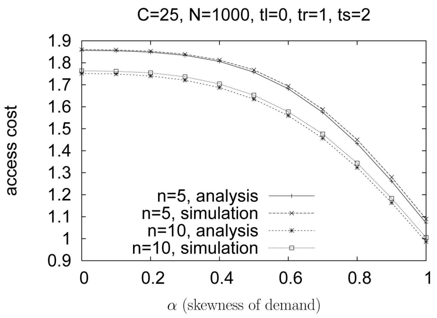


Fig. 3. Validation of the approximate analytical model in Section 3.2 through comparison with simulation results on the social cost of the group.

It is important to note that throughout this section, we use a number of settings to gain an understanding of state interaction in distributed caches and its consequences on local and group access costs. Most of these settings are *intentionally* very simple (that is, small "toy" examples) so that they can be possible to track.

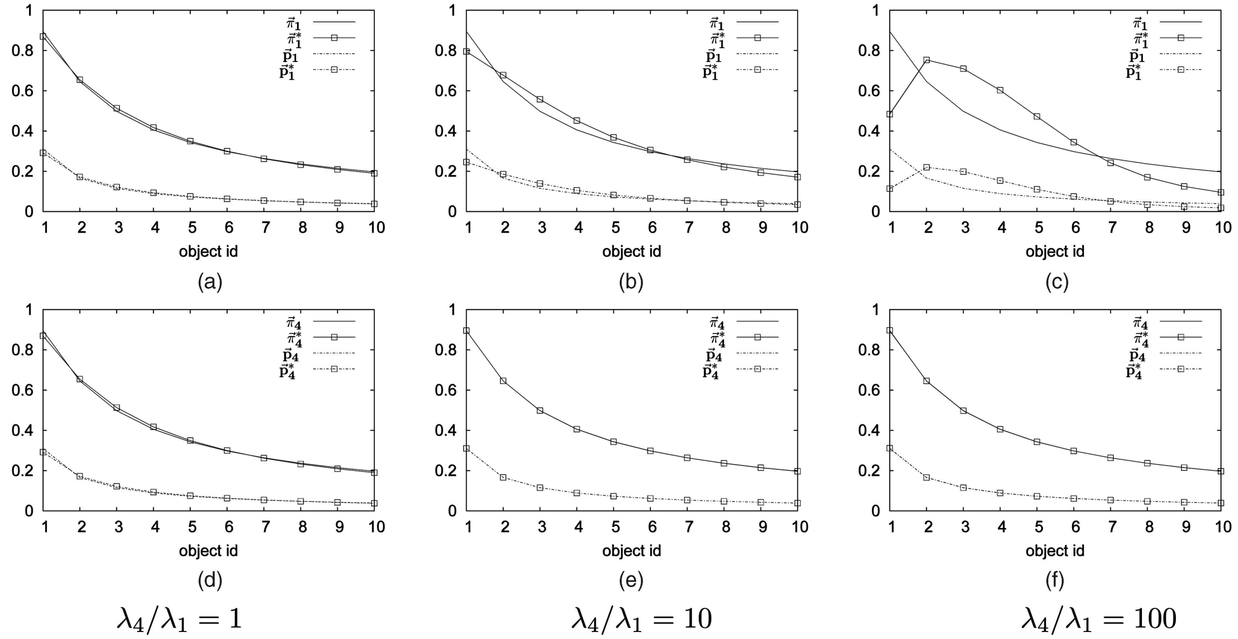
Also, it is important to note that the various parameterizations of our analytical and simulation models are not meant to represent particular content networking applications. Examining specific incarnations of the state interaction phenomenon is, after all, not our intention in this paper—which is the first to identify and analyze the problem. Rather, our exploration of the extent of mistreatment is meant to help us gain insights into the fundamental aspects of state interactions in distributed caching, such as its dependence on the request rate imbalance and the nodes' relative storage capacities.[6] In most of the following numerical results, we assume that nodes follow a common power-law demand distribution with skewness a as reported by several measurement studies [10]. We relax the common demand distribution assumption in Section 4.3, where we study the effect of nonhomogeneous demand on mistreatment and the social cost of the group. Overall, we pay greater attention to the case of homogeneous demand, since it is under such demand that cooperative caching becomes meaningful and effective (the benefits from employing cooperative caching diminish when the similarity of demand patterns is small).

4.1 Validation of the Analytic Model

The analytical model presented in Section 3.2 included a number of approximations: 1) the basic building block, the $LRU(\cdot,\cdot)$ function, is itself an approximation, and 2) the mapping of the effect of remote hits on the local state through (7) is approximate; the solution of the model through the iterative method is approximate.

In this section, we show that despite these approximations, the analytical model presented in Section 3.2 is able to produce fairly accurate results. We do so by comparing the model predictions with simulation results in Fig. 3. As evident from these results, the aforementioned approximations have a very limited effect on the model's prediction accuracy. We have obtained similar results across a wide variety of parameter sets. Thus, in the remainder of this section, we use this model to study several aspects of mistreatment due to state interaction.

[6] With respect to storage capacities, it is important to note that performance results depend on the relative size of the cache to the object space—that is, the ratio C/N, but not on the particular values of C and N, that is, our results are immune to scale.



(a) (b) (c)

(d) (e) (f)

$\lambda_4/\lambda_1 = 1$ $\lambda_4/\lambda_1 = 10$ $\lambda_4/\lambda_1 = 100$

Fig. 4. Analytical results on the effect of request rate imbalance on the per-object request and hit probabilities under LRU (values with "*" superscript) and LRU without state interaction. $\vec{p}$ denotes the demand and $\vec{\pi}$ the steady-state hit probabilities. Other parameters include $N = 10$, $n = 4$, $C = 4$, and $\alpha = 0.9$.

4.2 Understanding State Interaction

Fig. 4 provides a microscopic view of state interaction by showing its effect at the object level. The results are from an illustrative example involving a group of $n = 4$ nodes, each of which has storage for up to $C = 4$ objects in a universe of $N = 10$ objects (other parameters are shown in the caption and legends of the figure). Nodes $v_1, \dots, v_3$ have the same fixed request rate $\lambda_1 = 1$, whereas the overactive node v_4 has request rate $\lambda_4 = 1, 10, 100$ (that is, we have three sets of results that correspond to different λ_4, depicted in Figs. 4a and 4d, Figs. 4b and 4e, and Figs. 4c and 4f, respectively. Figs. 4a, 4b, and 4c depict the demand and the steady-state vector for node v_1 (which will be used as a representative for all three nonoveractive nodes), and Figs. 4d, 4e, and 4f depict the corresponding quantities for node v_4. Each graph includes four curves. The bottom two curves indicate the local demand distribution $\vec{p}$ and the aggregate demand distribution $\vec{p}^*$, which includes the effect of the other nodes' miss streams (each of these curves sums up to 1). The top two curves $\vec{\pi}$ and $\vec{\pi}^*$ show the steady-state vectors of a node when the input is $\vec{p}$ (no miss stream present) and $\vec{p}^*$ (miss stream present), respectively, as obtained from the analytical method in Section 3.2 (each of these curves sums up to C).

Looking at Figs. 4d, 4e, and 4f, we see that overactive node v_4 is not affected by the miss streams of other nodes. For $\lambda_4 = 10$ and 100, its aggregate demand and its steady state vector are identical to the corresponding ones without state interaction, that is, $\vec{p}_4^* \approx \vec{p}_4$ and $\vec{\pi}_4^* \approx \vec{\pi}_4$. For $\lambda_4 = 1$, there is a very slight effect due to the presence of the miss streams of the other three nodes, but this has almost no effect on the steady-state vector $\vec{\pi}_4^*$.

Looking at Fig. 4a, which corresponds to $\lambda_4 = 1$, we see that the same slight effect exists for node v_1 due to the reception of the other three miss streams. The situation, however, changes radically when increasing λ_4 (Figs. 4b and 4c). In that case, $\vec{p}_1^*$ and $\vec{p}_1$ and, as a consequence, $\vec{\pi}_1^*$ and $\vec{\pi}_1$ also become distinctively different. The intense miss stream from v_4 increases the popularity of some objects from the middle part of $\vec{p}_1$, thereby making them the most popular objects in $\vec{p}_1^*$. For example, when $\lambda_4 = 100$, objects 2, 3, and 4 become more popular than object 1. This change in the profile of $\vec{p}_1^*$ is then reflected in $\vec{\pi}_1^*$, thereby affecting its access cost (6), as we explain below.

4.3 Effect on Performance

Fig. 5 provides a macroscopic view of state interaction by considering its effects on the normalized access cost of each node. The normalized cost of node v_j under the aggregate demand $\vec{p}_j^*$ is defined as follows:

$$\hat{cost}_j(\vec{p}_j^*, \vec{p}_j) = \frac{cost_j(\vec{p}_j^*)}{cost_j^{iso}(\vec{p}_j)}, \tag{8}$$

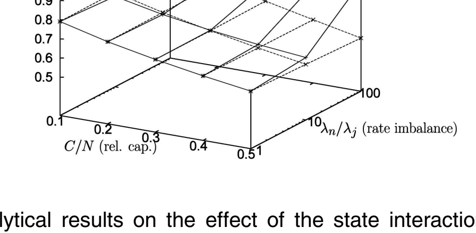


Fig. 5. Analytical results on the effect of the state interaction on the normalized access cost of the overactive node and the remaining nodes of the group under different relative storage capacities and request rate imbalances. No mistreatment occurs in this case.

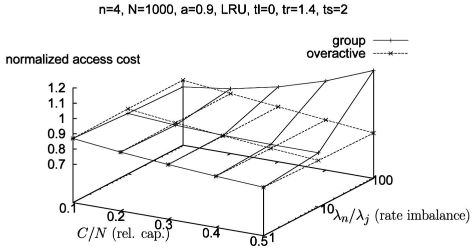


Fig. 6. Analytical results on the state interaction effect on the normalized access cost of the overactive node and the remaining nodes of the group under different relative storage capacities and request rate imbalances. Mistreatment occurs due to larger t_r.

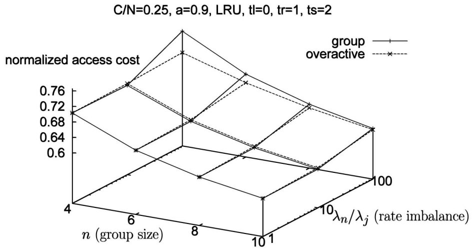


Fig. 7. Analytical results on the effect of the state interaction on the normalized access cost of the overactive node and the remaining nodes of the group under different group sizes and request rate imbalances.

where $cost_j^{iso}(\vec{p}_j) = \sum_{i=1}^{N} p_{ij} \cdot \left[\pi_{ij} \cdot t_l + (1 - \pi_{ij}) \cdot t_s\right]$ is the cost that would be incurred by v_j if it operated in isolation (outside the group) and received only its local demand $\vec{p}_j$. If $\hat{cost}_j < 1$, the node benefits from its participation in the group; otherwise, it is being mistreated. When considering two nodes, v_j and $v_{j'}$, then the fact that $1 > \hat{cost}_j > \hat{cost}_{j'}$ means that although both are better off by participating in the group, v_j gets a relatively larger benefit.

There are two main points to be concluded from Fig. 5. First, *it requires a very strong imbalance of request rates in order to create a substantial difference in the incurred normalized access costs*. In the presented example, the overactive node v_4 has a 30 percent reduction of its normalized cost, only when it produces a 100-fold more intense request stream. Even such a strong imbalance is not enough to mistreat the other nodes ($v_1, \ldots, v_3$ have normalized access costs < 1). For the occurrence of mistreatment, remote accesses have to be even more expensive (this is shown in Fig. 6, where t_r increases from 1 to 1.4, thereby making the normalized access cost of the group nodes > 1). Second, *the nodes must have a large storage capacity to be affected by state-interaction-related phenomena*. In the presented example, the nodes must have at least a 20 percent relative storage capacity C/N to be affected by the overactive node. Surprisingly, for small C/N, for example, less than 15 percent, the group nodes actually benefit more than the overactive node, that is, they achieve a smaller normalized access cost. In [17], we explained this peculiar phenomenon by arguing that the miss stream from the overactive node actually helps the other nodes in this case by creating more skewed demands for them, which lead to higher hit ratios. Fig. 7 shows that increasing the size of the group reduces the effects of the state interaction. This occurs as the miss stream of the overactive node(s) (here, just one) weakens by being divided across more nodes.

4.4 The Case of Nonhomogeneous Demand Patterns

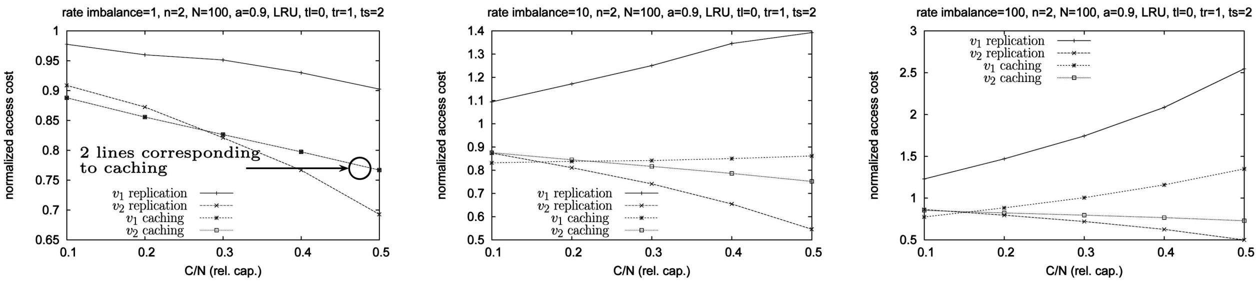
What we have described so far is fairly optimistic as we assumed that all participants in the distributed caching group exhibit similar access patterns. If this assumption does not hold, then the intensity of the mistreatment could be much higher, even for small C/N. To underscore this point, in this section, we will deviate from our course so far and examine mistreatments and the social cost of the group under nonhomogeneous demand distributions. For nonoveractive nodes, we will maintain the popularity ranking of objects as it was $(o_1, o_2, \ldots)$. For the overactive node, however, we will shift the popularity ranking by an *offset* O, $0 \leq O \leq N$, therefore making object $o_{1+(O+i-1) \bmod N}$ be the ith most popular object. We assign request probabilities taken from the same generalized power-law profile with skewness $a = 0.9$ that is used for the nonoveractive nodes. Fig. 9 depicts the demand distribution for the overactive node for $O = N/2 - 1$. The two graphs in Fig. 10 depict the normalized individual cost for the overactive and the nonoveractive nodes, as well as the social cost (normalization is obtained by dividing by the corresponding cost obtained when remote hits are not allowed to affect the local caching state). As is obvious, mistreatments can occur even under nonhomogeneous demand distributions. The concave profile with respect to O occurs as with high O; the popularity ranking starts to look like the original one due to "wrapping" after N. We have obtained similar results with several other perturbations of the popularity ranking [29].

4.5 Caching versus Replication

In this section, we will consider both replication and caching and compare their relative robustness to mistreatment. For replication, we will consider the SO replication algorithm of Leff et al. [20]. For simplicity of exposition and also to be able to compare with our previous numerical results in [18], we will consider a group with only $n = 2$ nodes and a universe of $N = 100$ objects. The three graphs in Fig. 8 depict the normalized access costs[7] for nodes v_1 and v_2 (overactive) for three cases of request imbalance: $1, 10,$ and 100. When there is no request imbalance (first graph), no node is mistreated. Caching yields the exact same performance for both nodes (the two curves for v_1 and v_2 are coinciding), whereas replication might unintentionally favor one of them (there are several optimal solutions, and the particular one chosen has to do with the specific solution algorithm that is employed; here, it is a linear programming (LP) relaxation of an integer problem solved via Simplex).

The different sensitivity to mistreatment becomes apparent as soon as request imbalance is introduced, that is, with $\lambda_2/\lambda_1 = 10$ and 100 (Figs. 8a and 8b). By observing these figures, we see that the curves for caching are always contained within the angle specified by the curves for replication (except for very small C/N, where we have the peculiar behavior of caching discussed in the previous section). The point to be kept from these results is that

[7] For the case of replication, the normalization is conducted by dividing with the performance of the GL replication strategy. See [18] for details.



(a)

(b)

(c)

Fig. 8. Analytical results on the comparison of replication and caching under three cases of request imbalance: (a) 1, (b) 10, and (c) 100.

replication is much more sensitive to mistreatment than caching. Under replication, the slightest imbalance of request intensities is directly reflected in the outcome of the replication algorithm. In contrast, the state interaction that takes place in caching is a much weaker catalyst for mistreatment. This fortunate weakness owes to the stochastic nature of caching and to the requirement for the concurrent occurrence of two independent events: An unpopular object must first be brought to the cache due to the local demand, and then, the miss stream must feed it with requests if it is to lock it in the cache (and thereafter beat the local request stream that tries to push it out and reclaim the storage space).

4.6 LRU versus LFU

Fig. 11 shows analytical results under LRU replacement, as well as simulation results under perfect LFU replacement [26] (two group sizes, $n = 2$ and $n = 4$, are considered). We plot the absolute instead of the normalized access costs, as we are considering different replacement algorithms. Looking first at LRU, we notice the following. The effects of state interaction (reflected in the width of the angle between group and overactive curves, after $\lambda_n/\lambda_j = 10$) decrease as the group grows larger, as also noted in the previous section. Moreover, the absolute access costs for both the group and the overactive node also decrease with the size of the group. The reason is that a bigger group has more aggregate storage capacity and thus succeeds in caching more distinct objects, which in turn benefits all the nodes.

Turning our attention to the LFU curves, we see a completely different behavior. For a given n, both the overactive node and the rest of the group have the same access cost, that is, *the request imbalance has no affect on the nodes under LFU*. This happens because once in a steady state, perfect LFU avoids replacement errors and thus does not give any opportunities for locking unpopular objects and losing storage due to the miss stream of remote overactive nodes. Thus, LFU has an advantage over LRU in terms of its immunity to request imbalance. What is even more interesting, however, is that the access cost under LFU remains the same under different n, that is, increasing the group size does not help in reducing the access costs. This happens because under LFU and common demand patterns, all the nodes end up caching exactly the same sets of objects. In such a group, a local miss is bound to miss also in the group. In other words, *LFU eliminates all the cooperation gain in groups of similar nodes*. This does not occur when the group operates under LRU: The replacement errors committed by the individual nodes in this case create a healthy amount of noise that increases the distinct objects held in the group, thereby decreasing the access cost of all the nodes. Thus, in large groups under small internode distances, LRU is more appropriate than LFU (see, for example, the access cost for small t_r in Fig. 12 in Section 5.1, where $n = 10$). When the internode distances increase, then the perfect ranking of objects under LFU becomes more important than the cooperation gain, and thus, LFU becomes better for the group (see Fig. 12 for large t_r).

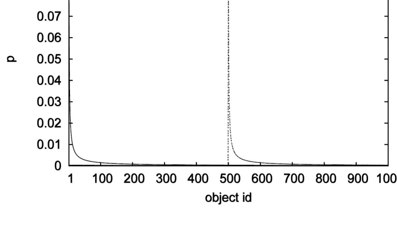


Fig. 9. The object demand distribution for the overactive node when the popularity ranking has been shifted by an offset O and the remaining nodes of the group.

4.7 L2 versus Non-L2 Caching

When a cache operates in L2 mode, it fetches and maintains a copy from the origin server for every request that it receives from a remote node (whether it hits or misses locally). In [17], we showed that *L2 caching eliminates the robustness to mistreatment of non-L2 caching, leading to a vulnerability level similar to the one under replication*. To understand this, one has to observe that in L2-caching and replication, locally irrelevant objects may occupy the local cache without the intervention of the local demand, whereas in non-L2 caching, the local demand has first to bring the objects in the cache and thus give other nodes the opportunity to maintain them there by feeding them with requests.

5 Mistreatment Due to the Use of a Common Scheme

In this section, we study cases of mistreatment due to the use of a common scheme vis-a-vis the object admission

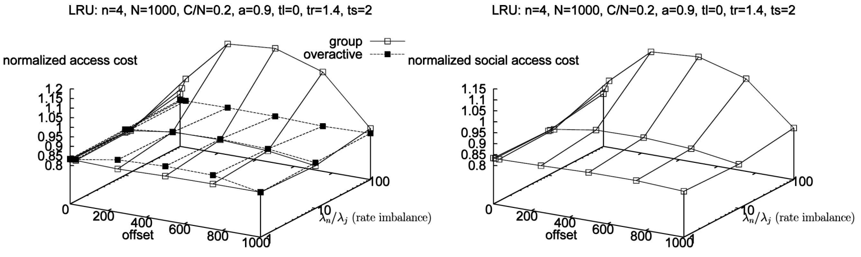


Fig. 10. Analytical results of the state interaction on the normalized access cost of the outlier and the remaining nodes in the group and the normalized social cost of the nodes in the caching group under nonhomogeneous demand distributions, where the overactive node's popularity ranking has been shifted rightwards by an offset O.

control algorithm. Specifically, we consider SC schemes like HASH and LRU-SC,[8] that is, schemes that allow for the existence of up to one copy of each object in the group, and MC schemes, that is, schemes that allow for the existence of multiple copies of the same object at different nodes of the group. All the replacement algorithms when combined with a non-SC object admission control fall into the MC category.

5.1 Single versus MC Schemes

Fig. 12 depicts simulation results showing the average access cost of a group (social cost) under different SC and MC schemes and for different values of t_r representing different levels of "tightness" of the group. Three types of demand are considered: lightly $(\mathrm{a} = 0.2)$, moderately $(\mathrm{a} = 0.6)$, and highly $(\mathrm{a} = 0.9)$ skewed demand. The following observations apply. *SC schemes* (that is, *HASH and LRU-SC, whose curves overlap almost completely in these figures, as the two have very similar caching behavior) perform better when the access cost between the nodes is small*. In such cases, the cost of local and remote accesses is similar, so it pays to eliminate multiple copies of the same object at different nodes and instead make room for storing a larger number of distinct objects. *MC schemes* (that is, *LRU and LFU) perform better when the access cost between the nodes is high*. In such cases, a much higher cost is incurred when an object is fetched from the group, so it becomes imperative to maintain some of the most popular objects locally (thereby creating multiple copies at different nodes). The threshold value of t_r at which the performance ranking between SC and MC changes depends on the skewness of the demand: the higher the skewness, the lower the value of t_r and the earlier the MC schemes become better.

It is also worthwhile noting that the curves for LFU are parallel to the x-axis, that is, the access cost is immune to the internode distance under LFU and identical demand. This happens because, as noted earlier, under LFU, all the nodes store the same objects, and this has the consequence of eliminating all remote hits. In that case, the exact value of the remote access cost does not affect the LFU curves, since there are no remote hits. Regarding the comparison between LRU and LFU, the figure shows that LFU is better when the remote access cost is high (see the discussion in Section 4.6 for an explanation of this).

The above observations highlight the fact that "fixed schemes" operate efficiently only under specific parameter sets. If these parameter sets are common to all the nodes, then good design choices can be made among the different schemes. However, when some of the parameters (for example, internode distances) are not common to all nodes, then *it may well be the case that no single scheme is appropriate for all the nodes. Enforcing a common scheme under such conditions is bound to mistreat some of the nodes.* The following section illustrates such an example.

5.2 Relaxing the Common Scheme Requirement

So far, we have assumed that all group nodes employ the same (common) caching scheme. In this section, we look at the advantages to be gotten from relaxing this constraint.

Consider the group depicted in Fig. 13 in which $n-1$ nodes are clustered together, meaning that they are very close to each other $(t_r \rightarrow t_l)$, while there is also a single "outlier" node at distance t'_r from the cluster. The $n-1$ nodes would naturally employ the LRU-SC scheme in order to capitalize on their small remote access cost. From the previous discussion, it should be clear that the best scheme for the outlier node would depend on t'_r. If $t'_r \rightarrow t_l$, the outlier should obviously follow LRU-SC and avoid duplicating objects that already exist elsewhere in the group. If $t'_r \gg t_l$, then the outlier should follow an MC scheme, for example, LRU.

To permit the outlier to adjust its caching behavior according to its distance from the group, we introduce the $\mathrm{LRU}(q)$ scheme under which objects that are fetched from

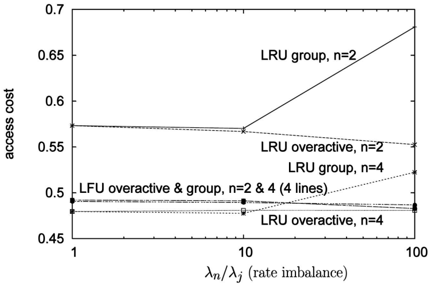
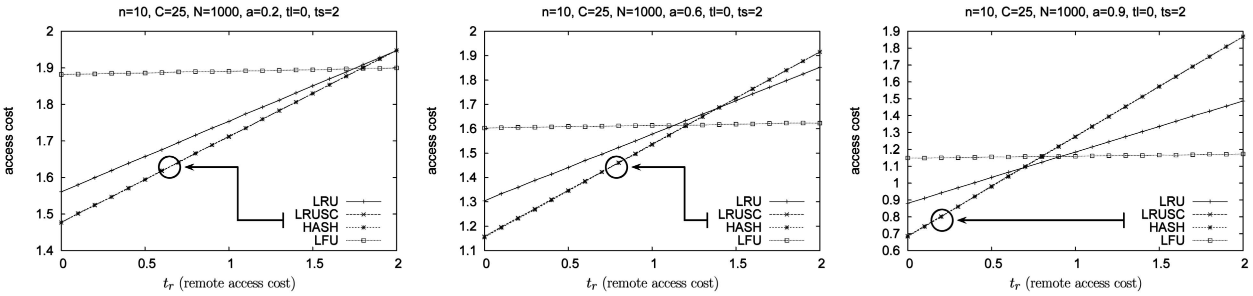


Fig. 11. Analytical results on the comparison LRU versus LFU.

[8] Under HASH, requests are received by the local node, which employs a hash function to identify the node that is responsible for the requested object. The responsible node returns the object immediately if it has already cached it or contacts the origin server and then returns it, also keeping a local copy in this case. The local node does not keep a local copy, unless it is the one responsible for that object according to the employed hash function. Under LRU-SC (single copy), a local copy is maintained at the local node only for objects that were fetched from the origin server. When an object is fetched from elsewhere in the group, no local copy is kept. In both cases, the number of copies of each object in the group is limited to at most one.



(a) (b) (c)

Fig. 12. Simulation results on the effect of the remote access cost t_r on the performance ranking of different SC and MC schemes for three cases of skewness of demand: (a) $a = 0.2$, (b) $a = 0.6$, and (c) $a = 0.9$. MC schemes (LRU and LFU) perform better when $t_r \rightarrow t_s$.

the origin server are automatically cached locally, but objects that are fetched from the group are cached locally only with probability q. For $q = 0$, $\mathrm{LRU}(q)$ reduces to LRU-SC, whereas for $q = 1$, it reduces to the MC LRU scheme. One may think of q as a *reliance parameter*, capturing the confidence that a node has in its ability to fetch objects efficiently (that is, "cheaply") from other members of the group.

Fig. 14 presents the performance of $\mathrm{LRU}(q)$ for $q = 0, 0.1, 0.5, 1$ under different t'_r. The results are normalized by dividing the access cost of each $\mathrm{LRU}(q)$ scheme by the corresponding access cost of the $\mathrm{LRU}(q = 1)$ scheme. The later can be seen as a basis for what a node can achieve by operating greedily, that is, when it always keeps a copy of each incoming object. Such a behavior corresponds to a node that wants to avoid relying on other nodes for fetching objects. As with the state interaction case, mistreatment is signified by a normalized access cost greater than 1.

Fig. 14 shows that for the considered scenario, always keeping local copies of all incoming objects (that is, employing LRU(1) and incurring a normalized access cost of 1) is a reasonably good choice across most values of t'_r. The only case in which LRU(1) performs poorly is when t'_r becomes very small, which corresponds to the case in which the node ceases to be an outlier and actually becomes part of the cluster. As discussed earlier, in this case, maintaining multiple object copies within the group becomes wasteful, with the optimal scheme being the SC LRU(0) scheme.

Another interesting observation from the above results is that there is a noticeable performance differential between the SC LRU(0) scheme and any other MC $\mathrm{LRU}(q)$ scheme with $q > 0$. A nonzero $\mathrm{LRU}(q)$ scheme, even one where q is small, is capable of eventually caching locally the most popular objects, even if this requires several misses. LRU(0), on the other hand, has almost no chance of bringing a globally popular object locally since it is much more likely for such an object to be cached in the cluster before being requested by the outlier node (which means that it will not be cached locally). When this happens for several popular objects, the performance degradation for the outlier node becomes very serious. That is why LRU(0) performs poorly for large values of t'_r.

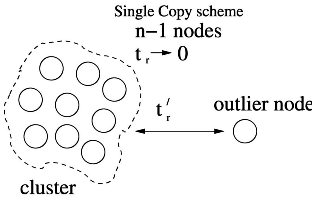


Fig. 13. An example of a group composed of a cluster of $n - 1$ nodes and a unique outlier.

6 Toward Mistreatment-Resilient Caching

From the exposition so far, it should be clear that there exist situations under which an inappropriate, or enforced, scheme may mistreat some of the nodes. Although we have focused on detecting and analyzing two causes of mistreatment which appear to be important (namely, due to cache state interactions and the adoption of a common cache management scheme), it should be evident that mistreatments may well arise through other causes. For example, we have not investigated the possibility of mistreatment due to request rerouting [25], not to mention that there are vastly more parameter sets and combinations of schemes that cannot all be investigated exhaustively.

To address the above challenges, we first sketch a general framework for designing mistreatment-resilient schemes. We then apply this general framework to the two types of mistreatments that we have considered in this work. We target "open systems" in which group settings (for example, number of nodes, distances, and demand patterns) change dynamically. In such systems, it is not possible to address the mistreatment issue with predefined, fixed designs (for example, using the results of the previous section for

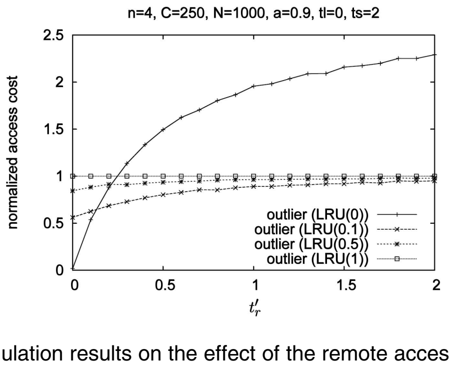


Fig. 14. Simulation results on the effect of the remote access cost t'_r on the normalized access cost of the outlier node under different $\mathrm{LRU}(q)$ schemes.

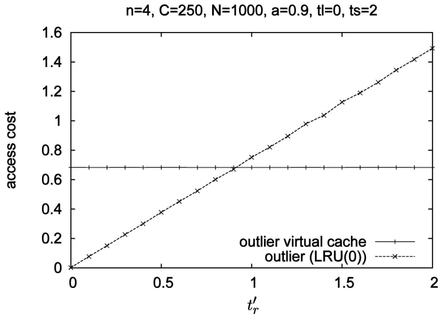


Fig. 15. Simulation results on the effect of the remote access cost t_r' on the access cost of the outlier node under the virtual cache and LRU(0) schemes.

selecting a fixed value for the reliance parameter q). Instead, we believe that *nodes should adjust their scheme dynamically so as to avoid or respond to mistreatment if and when it emerges*. To achieve this goal, we argue that the following three requirements are necessary:

Detection Mechanism. This requirement is obvious but not trivially achievable when operating in a dynamic environment. *How can a node realize that it is being mistreated?* In our previous work on replication [18], a node compared its access cost under a given replication scheme with the guaranteed maximal access cost obtained through GL replication. This gave the node a "reference point" for a mistreatment test. In that game-theoretic framework, we considered nodes that had a priori knowledge of their demand patterns and thus could easily compute their GL cost thresholds. In caching, however, demand patterns (even local ones) are not known a priori, nor are they stationary. Thus, in our DSC setting, the nodes have to estimate and update their thresholds in an online manner. We believe that a promising approach for this is *emulation*. Fig. 16 depicts a node equipped with an additional *virtual cache*, alongside its "real" cache that holds its objects. The virtual cache does not hold actual objects but rather object identifiers. It is used for emulating the cache contents and the access cost under a scheme *different from* the one being currently employed by the node to manage its "real" cache under the same request sequence (notice that the input request stream is copied to both caches). The basic idea is that *the virtual cache can be used for emulating the threshold cost that the node can guarantee for itself by employing a greedy scheme*.

Mitigation Mechanism. This requirement ensures that a node has a mechanism that allows it to react to mistreatment —a mechanism via which it is able to respond to the onset of mistreatment. In the context of the common scheme problem, the ability to adjust the *reliance parameter* q acted as such a mechanism. In the context of the state interaction problem, one may define an *interaction parameter* p_s and the corresponding $\mathrm{LRU}(p_s)$ scheme in which a remote hit is allowed to affect the local state with probability p_s, whereas it is denied such access with probability $(1 - p_s)$. As will be demonstrated later on, nodes may avoid mistreatment by selecting appropriate values for these parameters according to the current operating conditions.

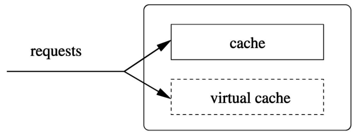


Fig. 16. Block diagram of a node equipped with a virtual cache.

Control Scheme. In addition to the availability of a mistreatment mitigation mechanism (for example, $\mathrm{LRU}(q)$), there needs to be a programmatic scheme for adapting the control variable(s) of that mechanism (for example, how to set the value of q). Since the optimal setting of these control variables depends heavily on a multitude of other time-varying parameters of the DSC system (for example, group size, storage capacities, demand patterns, and distances), it is clear that there cannot be a simple (static) rule of thumb for optimally setting the control variables of the mitigation mechanism. To that end, dynamic feedback-based control becomes an attractive option.

To make the previous discussion more concrete, we now focus on the common scheme problem and demonstrate a mistreatment-resilient solution based on the previous three principal requirements. A similar solution can be developed for the state interaction problem.

6.1 Resilience to Common-Scheme-Induced Mistreatments under "Soft Selfishness"

We start with a simple "binary" policy that allows a node to change operating parameters by selecting between two alternative schemes. This can be achieved by using the virtual cache for emulating the LRU(1) scheme, that is, the scheme in which the reliance parameter q is equal to 1 (capturing the case that the outlier node does not put any trust on the remote nodes for fetching objects and thus keeps copies of all incoming objects after local misses). Equipped with such a device, the outlier can calculate a running estimate of its threshold cost based on the objects it emulates as present in the virtual cache.[9] By comparing the access cost from sticking to the current scheme to the access cost obtained through the emulated scheme, the outlier can decide which one of the two schemes is more appropriate. For example, it may transit between the two extreme $\mathrm{LRU}(q)$ schemes—the $\mathrm{LRU}(q = 0)$ scheme and the $\mathrm{LRU}(q = 1)$ scheme. Fig. 15 shows that the relative performance ranking of the two schemes depends on the distance from the group t_r' and that there is a value of t_r' for which the ranking changes.

The above mechanism is appropriate for avoiding mistreatments as defined for "soft-selfish" nodes that are satisfied if they can guarantee a performance level at least as good as the one they can guarantee for themselves in isolation. In the sequel, we sketch a scheme for addressing "hard-selfish" nodes, that is, nodes that want to minimize their cost granted the current behavior by other nodes. Such a goal is clearly more ambitious than just avoiding

[9] The outlier can include in the emulation the remote fetches that would result from misses in the emulated cache contents; this would give it the exact access cost under the emulated scheme. A simpler approach would be to disregard the remote fetches and thus reduce the internode query traffic; this would give it an upper bound on the access cost under the emulated scheme.

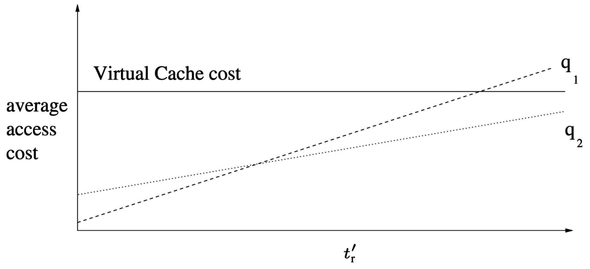


Fig. 17. Representative development of the average object access cost as a function of the reliance parameter and distance of the outlier from the cluster.

mistreatment. In game-theoretic terms, it amounts to selecting a "best response" strategy.

6.2 Resilience to Common-Scheme-Induced Mistreatments under "Hard Selfishness"

Hard selfishness demands setting the reliance parameter q so as to minimize the local cost under the current behavior by the other nodes. Indeed, there are situations in which intermediate values of q, $0 < q < 1$, are better than both $q = 0$ and $q = 1$, which is what the previous binary scheme allows (see the LRU(0.1) and LRU(0.5) curves in Fig. 14). Consider two different values of the reliance parameter q_1 and q_2 such that $q_1 < q_2$. Fig. 17 illustrates a typical development of the average object access cost under q_1 and q_2 as a function of the distance t'_r of the outlier node from its cooperative cluster. As discussed in the previous section, q_1 (q_2) will perform better with small (large) t'_r.

In a longer version [16] of this article, we propose and analyze a Proportional-Integral-Differential (PID) controller for controlling the value of q so as to achieve the aforementioned best response. A brief sketch of its operation follows: The controller maintains running averages of the actual cost under the current q and the corresponding one from a virtual cache that emulates an $\mathrm{LRU}(q = 1)$ scheme. The controller changes the actual q so as to minimize the actual cost. In doing so, it uses as point of reference the emulated cost (see [16] for details). In the same document, we argue that such a controller can be realized with minimal overhead in terms of information exchange, required memory, and processing.

Performance Evaluation. In order to evaluate our adaptive scheme, we compare its cumulative average access cost to the corresponding cost of one of the two extreme static schemes $(\mathrm{LRU}(q = 0), \mathrm{LRU}(q = 1))$. Thus, we define the following performance metric:

$$minimum\ cost\ reduction\ (\%) = 100 \cdot \frac{cost_{static} - cost_{adaptive}}{cost_{static}}, \tag{9}$$

where $cost_{adaptive}$ is the access cost of our adaptive mechanism and $cost_{static}$ is the minimum cost among the two static schemes:

$$cost_{static} = min(cost(LRU(q = 0), LRU(q = 1)).$$

This metric captures the minimum additional benefit that our adaptive scheme has over the previous static schemes. To capture the maximum additional benefit of our adaptive scheme (the optimistic case), we similarly define the *maximum cost reduction* as in (9), where

$$cost_{static} = max(cost(LRU(q = 0), LRU(q = 1)).$$

We evaluate the performance of our PID-style feedback controller experimentally by considering a scenario in which the distance between the outlier node and the cooperative group (t'_r) changes according to the Modified Random Waypoint Model [21]. The motivation for such a scenario comes from a wireless caching application [32]. A detailed description of the design of this experiment can be found in [16]. Fig. 18 summarizes results we obtained under different cache sizes, demand skewness, and movement speed $V_{max} = 1$ distance units/time unit (similar results are observed under higher speeds as well). All experiments were repeated 10 times, and we include 95th-percentile confidence intervals in the graphs.

By employing our adaptive scheme, the outlier achieves a maximum cost reduction that can be up to 60 percent under skewed demand. The depicted profile of the maximum cost reduction curve can be explained as follows:

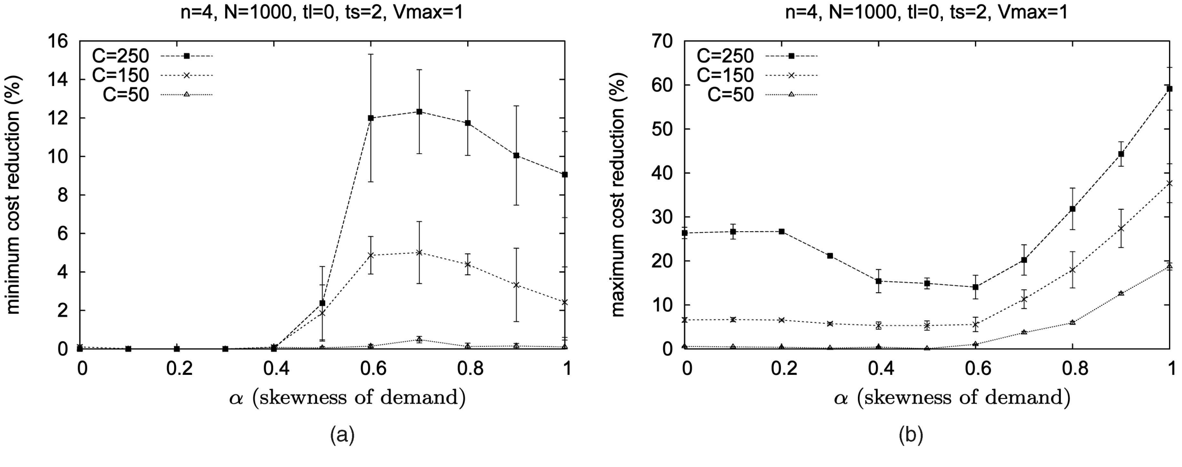


Fig. 18. Simulation results on the cost reduction that is achieved using our adaptive mechanism. (a) The minimum cost reduction. (b) The maximum cost reduction.

The worst performance of the static schemes appears at the two extremes of skewness. Under uniform demand, $a = 0$, we get the worst performance of the LRU(1) static scheme, whereas under highly skewed demand, $a = 1$, we get the worst performance of the LRU(0) static scheme. In the intermediate region, both static schemes provide for some level of compromise, and thus, the ratio of the cost achieved by either scheme to the corresponding cost of the adaptive scheme becomes smaller than in the two extremes.

Turning our attention to the minimum cost reduction, we observe that it can be substantial under skewed demand and disappears only under uniform demand (such demand, however, is not typically observed in measured workloads [2]). The explanation of this behavior is as follows: At the two extreme cases of skewness, one of the static scheme reaches its optimal performance—under low skewed demand, the best static scheme is the LRU(0), and under high skewed demand, the best static scheme is the LRU(1). Thus, the ratio of the cost achieved by the best static scheme and the corresponding cost of our adaptive scheme gets maximized in the intermediate region in which neither of the static schemes can reach its best performance.

6.3 Resilience to State-Interaction-Induced Mistreatments

Immunizing a node against mistreatments that emerge from state interactions could be similarly achieved. The interaction parameter p_s can be controlled using schemes similar to those we considered above for the reliance parameter q. It is important to note that one may argue for *isolationism* (by permanently setting $p_s = 0$) as a simple approach to avoid state-interaction-induced mistreatments. This is not a viable solution. Specifically, by adopting an $\mathrm{LRU}(p_s = 0)$ approach, a node is depriving itself from the opportunity of using miss streams from other nodes to improve the accuracy of LRU-based cache/no-cache decisions (assuming a uniform popularity profile for group members). This was highlighted in the results shown in Fig. 5.

To conclude this section, we note that the approaches we presented above for mistreatment resilience may be viewed as “passive” or “end to end” in the sense that a node infers the onset of mistreatment *implicitly* by monitoring its utility function. As we alluded at the outset of this paper, for the emerging class of network applications for which grouping of nodes is “ad hoc” (that is, not dictated by organizational boundaries or strategic goals), this might be the only realistic solution. In particular, to understand “exactly how and exactly why” mistreatment is taking place would require the use of proactive measures (for example, monitoring/policing group member behaviors, measuring distances with pings, and so forth), which would require group members to subscribe to some common services or to trust some common authority—both of which are not consistent with the autonomous nature (and the mutual distrust) of participating nodes.

7 Summary and Concluding Remarks

Distributed on-demand caching enables loosely coupled groups of nodes to share their (storage) resources to achieve higher efficiencies and scalability. In addition to its traditional use in content distribution/delivery networks, distributed caching is also used as an important building block of many emerging applications and protocols, including its use in route caching in ad hoc networks [24] and in P2P content replication [6], [13].

Summary. This paper has uncovered the susceptibility of nodes participating in a distributed on-demand caching group to being *mistreated*. We have identified two causes of mistreatments, namely, mistreatment due to cache *state interactions* between various members of the group and mistreatment due to the use of a *common scheme* for cache management across all members of the group. We have backed up our findings by analytic models, numerical solutions of these models, and simulations in which assumptions (necessary for analysis) have been relaxed.

The results of our analysis and evaluation suggest that on-demand distributed caching is fairly resilient to the onset of mistreatment as long as proxying (L2 caching) is not enabled and as long as intragroup access costs do not include outliers. More constructively, we have outlined an efficient emulation-based approach that allows individual nodes to decide autonomously (that is, without having to trust any other node or service) whether they should stick to or secede from a caching group, based on whether or not their participation is beneficial to their performance compared to a selfish greedy scheme.

Other Incarnations of Mistreatment in Online Distributed Resource Management Problems. In this paper, we focused on distributed caching as an instance of an online protocol for the management of a distributed resource, namely, the limited storage available at each node. Although our exposition has focused on the well-known problem of caching “retrievable” content (for example, Web pages and media objects), it should be evident that our results extend to *any* other type of cached content, including nonretrievable content used as part of the control plane of a distributed protocol or application (for example, route paths stored in routing tables of group members). Clearly, given the different nature of the workloads that such distributed resources must support, a more specific examination of potential mistreatments in such settings is warranted and is a current subject of our inquiry.

Coincidental versus Adversarial Mistreatment. In this paper, we focused on the onset of mistreatment due to benign operating conditions of a caching group. For instance, we identified rate imbalance (of local versus remote requests streams) conditions, as well as cache sizing conditions, that are necessary for mistreatment to occur. As such, the cases of mistreatment we have uncovered could be considered “coincidental.” Another possible source of mistreatment, however, could be adversarially motivated in the sense that one (or more) of the group members collude to negatively impact the performance of other members. Although we did not consider adversarial mistreatments *per se*, our results suggest that distributed caching is fairly immune to *high potency exploits* [11] (low-rate attacks) by nonclairvoyant adversaries. More work is needed to characterize the vulnerability of distributed caching to more elaborate adversarial exploits, including those from more powerful agents (for example, those with knowledge of a victim’s cache contents).

ACKNOWLEDGMENTS

Azer Bestavros and Ibrahim Matta are supported in part by a number of US National Science Foundation (NSF) awards, including CNS Cybertrust Award No. 052447, CNS NeTS Award No. 0520166, CNS ITR Award No. 0205294, and EIA RI Award No. 0202067. Ioannis Stavrakakis is supported in part by EU IST projects CASCADAS and CONTENT. Nikolaos Laoutaris is supported by a Marie Curie Outgoing International Fellowship of the EU MOIF-CT-2005-007230.

REFERENCES

[1] M.F. Arlitt and C.L. Williamson, "Web Server Workload Characterization: The Search for Invariants," *Proc. ACM Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '96),* pp. 126-137, 1996.

[2] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. INFOCOM '99,* Mar. 1999.

[3] J.W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery across Adaptive Overlay Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 5, pp. 767-780, Oct. 2004.

[4] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C.H. Papadimitriou, and J. Kubiatowicz, "Selfish Caching in Distributed Systems: A Game-Theoretic Analysis," *Proc. ACM Symp. Principles of Distributed Computing (PODC '04),* July 2004.

[5] E.G. Coffman and P.J. Denning, *Operating Systems Theory.* Prentice Hall, 1973.

[6] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM '02,* Aug. 2002.

[7] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," *Proc. ACM Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '90),* pp. 143-152, 1990.

[8] D. Wessels and K. Claffy, *ICP and the Squid Web Cache,* http://www.ircache.net/~wessels/Papers/icp-squid.ps.gz, 2007.

[9] Ö. Erçetin and L. Tassiulas, "Market-Based Resource Allocation for Content Delivery in the Internet," *IEEE Trans. Computers,* vol. 52, no. 12, pp. 1573-1585, Dec. 2003.

[10] L. Fan, P. Cao, J. Almeida, and A.Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Trans. Networking,* vol. 8, no. 3, pp. 281-293, 2000.

[11] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources," *Proc. 12th IEEE Int'l Conf. Network Protocols (ICNP '04),* Oct. 2004.

[12] S. Jin and A. Bestavros, "Sources and Characteristics of Web Temporal Locality," *Proc. Eighth IEEE/ACM Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '00),* Aug. 2000.

[13] J. Kangasharju, K.W. Ross, and D.A. Turner, *Optimal Content Replication in P2P Communities,* manuscript, 2002.

[14] R. Landry and I. Stavrakakis, "Queueing Study of a 3-Priority Policy with Distinct Service Strategies," *IEEE/ACM Trans. Networking,* vol. 1, no. 5, pp. 576-589, 1993.

[15] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD Interconnection of LRU Caches and Its Analysis," *Performance Evaluation,* vol. 63, no. 7, pp. 609-634, 2006.

[16] N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, "Distributed Selfish Caching," Technical Report BUCS-TR-2006-003, Computer Science Dept., Boston Univ., http://www.cs.bu.edu/techreports/2006-003-distributed-selfish-caching.ps.Z, Feb. 2006.

[17] N. Laoutaris, G. Smaragdakis, A. Bestavros, and I. Stavrakakis, "Mistreatment in Distributed Caching Groups: Causes and Implications," *Proc. INFOCOM '06,* Apr. 2006.

[18] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed Selfish Replication," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 12, pp. 1401-1413, 2006.

[19] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "On the Optimization of Storage Capacity Allocation for Content Distribution," *Computer Networks,* vol. 47, no. 3, pp. 409-428, Feb. 2005.

[20] A. Leff, J.L. Wolf, and P.S. Yu, "Replication Algorithms in a Remote Caching Architecture," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 11, pp. 1185-1204, Nov. 1993.

[21] G. Lin, G. Noubir, and R. Rajaraman, "Mobility Models for Ad Hoc Network Simulation," *Proc. INFOCOM '04,* Mar. 2004.

[22] T. Loukopoulos, P. Lampsas, and I. Ahmad, "Continuous Replica Placement Schemes in Distributed Systems," *Proc. 19th ACM Int'l Conf. Supercomputing (ICS '05),* June 2005.

[23] A. Mahanti, C. Williamson, and D. Eager, "Traffic Analysis of a Web Proxy Caching Hierarchy," *IEEE Network,* vol. 14, no. 3, pp. 16-23, May 2000.

[24] M. Marina and S. Das, "Performance of Route Caching Strategies in Dynamic Source Routing," *Proc. Int'l Workshop Wireless Networks and Mobile Computing (WNMC '01) in conjunction with Int'l Conf. Distributed Computing Systems (ICDCS '01),* 2001.

[25] J. Pan, Y.T. Hou, and B. Li, "An Overview DNS-Based Server Selection in Content Distribution Networks," *Computer Networks,* vol. 43, no. 6, Dec. 2003.

[26] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Computing Surveys,* vol. 35, no. 4, pp. 374-398, 2003.

[27] K. Psounis, A. Zhu, B. Prabhakar, and R. Motwani, "Modeling Correlations in Web Traces and Implications for Designing Replacement Policies," *Computer Networks,* vol. 45, July 2004.

[28] K.W. Ross, "Hash-Routing for Collections of Shared Web Caches," *IEEE Network,* vol. 11, no. 6, Nov. 1997.

[29] G. Smaragdakis, "Notes on the Effect of Different Access Patterns on the Intensity of Mistreatment in Distributed Caching Groups," Technical Report BUCS-TR-2006-023, Computer Science Dept., Boston Univ., http://www.cs.bu.edu/techreports/2006-023-effect-different-patterns.ps, Sept. 2006.

[30] X. Tang and S.T. Chanson, "Adaptive Hash Routing for a Cluster of Client-Side Web Proxies," *J. Parallel and Distributed Computing,* vol. 64, no. 10, pp. 1168-1184, Oct. 2004.

[31] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H.M. Levy, "On the Scale and Performance of Cooperative Web Proxy Caching," *ACM SIGOPS Operating Systems Rev.,* vol. 33, no. 5, pp. 16-31, 1999.

[32] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *Proc. INFOCOM '04,* 2004.

Nikolaos Laoutaris received the BSc degree in computer science in 1998, the MSc degree in telecommunications and computer networks in 2001, and the PhD degree in 2004 for his work in the area of content networking from the Department of Informatics and Telecommunications, University of Athens, Greece. His main research interests are in the analysis of algorithms and the performance evaluation of Internet content distribution systems (content delivery network (CDN), peer-to-peer (P2P), Web caching) and multimedia streaming applications. He is currently a Marie Curie outgoing international postdoctoral fellow at the Computer Science Department of Boston University.

Georgios Smaragdakis received the diploma in electronic and computer engineering from the Technical University of Crete, Greece. He is currently working toward the PhD degree in computer science at Boston University, Massachusetts. His research interests include the design and analysis of network systems with main applications in overlay network creation and maintenance, and resource allocation and sharing.

Azer Bestavros received the PhD degree in computer science from Harvard University in 1992. He is a professor and the chair of the Computer Science (CS) Department at Boston University. His research interests are in networking and real-time systems. His seminal works include his generalization of classical rate-monotonic analysis to accommodate probabilistic guarantees, his pioneering of the push model for Internet content distribution adopted years later by content delivery networks (CDNs), and his characterization of Web traffic self-similarity and reference locality. His research work has culminated so far in 10 PhD theses, more than 80 masters and undergraduate student projects, five US patents, and two startup companies. With more than 3,000 citations, CiteSeer ranks him in the top 5 percent of its list of 10,000 most cited authors, and since 1999, WebBib has ranked his publications as constituting one of the top three bodies of Web-related research by a single author. His research has been funded by more than $15 million of government and industry grants. He has served as the general chair, program committee (PC) chair, officer, or PC member of most major conferences in networking and real-time systems. He has received distinguished service awards from both the ACM and the IEEE and is a distinguished speaker of the IEEE.

Ibrahim Matta received the PhD degree in computer science from the University of Maryland, College Park, in 1995. He is an associate professor of computer science at Boston University. His research involves routing and transport protocols, focusing on resiliency and safety aspects. He has published more than 70 refereed technical papers and was guest coeditor of three special journal issues. He received the US National Science Foundation Faculty Early Career Development (CAREER) Award in 1997. He is on the editorial board of the *Computer Networks Journal*. He was the general chair of the Fourth International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2006), technical program cochair of the 13th IEEE International Conference on Network Protocols (ICNP 2005), technical program cochair of the Third International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics 2005), Internet cochair of INFOCOM 2005, publication chair of INFOCOM 2003, and tutorial and panel chair of Hot Interconnects 2001. He was co-organizer and technical program cochair of the EU-US NeXtworking 2003. He is a senior member of the IEEE.

Ioannis Stavrakakis received the diploma in electrical engineering from the Aristotelian University of Thessaloniki, Greece, in 1983, and the PhD degree in electrical engineering from the University of Virginia, in 1988. He was an assistant professor in computer science and electrical engineering (CSEE), University of Vermont, during 1988-1994, was an associate professor of electrical and computer engineering (ECE), Northeastern University, Boston, during 1994-1999, was an associate professor of informatics and telecommunications at the University of Athens, Greece, during 1999-2002, and has been a professor since 2002. His teaching and research interests are focused on resource allocation protocols and traffic management for communication networks, with recent emphasis on peer-to-peer, wireless, sensor, and ad hoc networking. His past research has been published in more than 130 scientific journals and conference proceedings and was funded by the US National Science Foundation (NSF), DARPA, General Telephone and Electronics (GTE), BBN, and Motorola (USA), as well as Greek and European Union Information Society Technologies (IST) Funding agencies. He has served repeatedly in NSF and IST research proposal review panels and been involved in the organization of numerous conferences sponsored by the IEEE, the ACM, ITC, and IFIP societies. He is a fellow of the IEEE, a member of (and has served as an elected officer of) the IEEE Technical Committee on Computer Communications (TCCC), and is the chairman of IFIP WG6.3. He has served as a coorganizer of the 1996 International Teletraffic Congress (ITC) Mini-Seminar, the organizer of the 1999 IFIP WG6.3 Workshop, a technical program cochair for the IFIP Networking 2000 Conference, Fifth European Wireless Conference (EWC 2004), and IFIP Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2005), the vice-general chair for the IFIP Networking 2002 Conference, the organizer of the COST-IST(EU)/NSF(USA)-sponsored NeXtworking 2003, and the Workshop on Autonomic Communications (WAC 2005). He is an associate editor for the *IEEE/ACM Transactions on Networking*, the *ACM/Baltzer Wireless Networks Journal*, and the *Computer Networks Journal*.