Towards A Scalable Information-Centric Approach to Cache Management

Junaid Ahmed Khan
University of Memphis
Memphis, TN, USA
junaid.khan@memphis.edu

Cedric Westphal
Huawei Technology
Santa Clara, CA, USA
cwestphal@gmail.com

J.J. Garcia-Luna-Aceves
University of California, Santa Cruz, CA, USA
jj@soe.ucsc.edu

Yacine Ghamri-Doudane
L3i Lab, University of La Rochelle, France
yacine-ghamri@univ-lr.fr

ABSTRACT
All Information-Centric Networking (ICN) architectures proposed to date aim at connecting users to content directly, rather than connecting clients to servers. Surprisingly, however, although content caching is an integral of any information-Centric Network, limited work has been reported on information-centric management of caches in the context of an ICN. Indeed, approaches to cache management in networks of caches have focused on network connectivity rather than proximity to content.

We introduce the Network-oriented Information-centric Centrality for Efficiency (NICE) as a new metric for cache management in information-centric networks. We propose a method to compute information-centric centrality that scales with the number of caches in a network rather than the number of content objects, which is many orders of magnitude larger. Furthermore, it can be pre-processed offline and ahead of time. We apply the NICE metric to a content replacement policy in caches, and show that a content replacement based on NICE exhibits better performances than LRU and other policies based on topology-oriented definitions of centrality.

KEYWORDS
ICN, Cache Management, Graph Centrality, Content Offloading

1 INTRODUCTION
Under the pressure of rapidly increasing bandwidth demands, Internet Service Providers (ISP) have been trying to offload traffic away from their networks onto local caches, either co-located with access points of WiFi networks, base stations of small cells, or provided by end users in an ad-hoc fashion. The nodes around the end-user form a network from which the consumer can retrieve the content opportunistically. However, selecting what content to cache at these nodes is a complex problem: while the most popular content should be available to most users, it is expected that there will be some redundancy in the caches that are reachable by a given user. Indeed, putting the most popular content in all caches (as a cache replacement policy such as LFU or LRU attempts to emulate) would result in excessive redundancy and a waste of caching capacity.

It has been known [1, 2] since the days of web caching that the coordination of the caches yields better performance. However, as Section 2 elaborates, prior work on networks of caches has focused primarily on caching policies in which each cache makes decisions independently of others or its network placement, or policies that take into account the network connectivity of caches. Centrality [3] is a concept from graph theory typically applied to social networks. It is used to find important nodes in a graph. A high centrality score reflects a high topological connectivity for a node in the network. Typical centrality measures are: degree (the number of directly connected nodes), closeness (the average length of the shortest paths between the node and all other nodes in the graph); betweenness (the number of shortest paths between all pairs of nodes in the graph going through
a specific node), and eigenvector centrality (a measure of node influence in the network). Recent work on networks of caches has used the centrality of caches for content placement [4] [5]. However, using centrality solely based on a graph representing the topology of a network fails to address the fact that the primary role of caches in an information-centric network (ICN) is to bring users and content closer to each other, rather than ensuring that a user is close to a specific node or type of node.

The main contribution of this paper is the introduction of an information-centric approach to cache management that scales with the number of caching sites rather than the number of content objects, which would not be feasible.

Section 3 introduces the **Network-oriented Information-centric Centrality for Efficiency (NICE)** as a metric of centrality for caches that focuses on the main objective of an ICN, which is to bring users closer to content, while taking into account the location of caches and content in a network. Equally important, we introduce a scalable method to calculate NICE without a priori knowledge of the content placement, based only upon pre-computed combinations of caches holding a specific content object. Expanding on the definition of betweenness centrality, for a cache $c$, its NICE betweenness is calculated as a function of the number of shortest paths that include $c$ for content $x$, the popularity of content $x$, and the number of shortest paths from the users to content $x$. We also consider a NICE closeness. These are defined more formally in Section 3.

Section 4 demonstrates the utility of the NICE metric by presenting a cache replacement policy and algorithm that take into account the NICE value of a node to update its cache. Namely, we suggest to replace content only if it increases the NICE value of the cache.

Section 5 presents the results of our evaluation of the proposed NICE-based cache replacement policy by simulations using a large number of scenarios. The results show that the proposed NICE-based cache replacement outperforms typical centrality schemes, or schemes without coordination. Section 6 concludes the paper along insights into future directions.

## 2 RELATED WORK

Content caching has been studied for some time by the research community, spanning a wide spectrum, including small cell networks (SCNs) [6, 7], content distribution networks (CDNs) [8] and information-centric networks (ICN) [9]. For example, Sourlas et al. [10] study making distributed cache management decisions in order to efficiently place replicas of information in dedicated storage devices attached to nodes in an ICN. Similarly, Wang et al. [11] address the distribution of the cache capacity across routers under a constrained total storage budget for the network. The authors found that network topology and content popularity are two important factors that affect where exactly cache capacity should be placed. Dabirmoghaddam et al. [12] and Fayazbakhsh et al. [13] argue through analytical models and experiments that caching content should be done at caches near consumers; however, these works assume that caching decisions are made by each cache independently of others.

Several works [14–16] consider the joint routing and caching problem. The goal is to place the content in the cache on the path, so that the cache is reachable by users for a wider range of content. Rossini and Rossi [17] consider the intersection of caching policy within a node, of meta-caching between nodes, and the joint impact of routing, to demonstrate a significant improvement when meta-caching (whether or not to consider an object for caching based upon the other caches) and routing decisions are tightly coupled.

Yu et al. [18] considered on-path caching as a method to increase path capacity, while Ramakrishnan et al. [19] devised a content placement mechanism which improve on the joint utilization of a set of caches.

Pantazopoulos et al. [20] define a “conditional betweenness centrality” and uses this metric to choses which nodes will cache the content. The Socially-Aware Caching Strategy (SACS) [21] for content-centric networks (CCN) uses social information in order to privilege Influential users in the network by pro-actively caching the content they produce. The authors detect the influence of users within a social network by using the Eigenvector and PageRank centrality measures.

Rossi et al. [22] present a caching approach for ICNs in which the sizing of the content store is based upon centrality. The authors exploit different centralities (betweenness, closeness, stress, graph, eccentricity and degree) to allocate content storage to nodes. It is proposed that a simple degree centrality-based allocation is sufficient to allocate content storage capacity. Similarly, Chai et al. [23] show that a higher cache-hit rate can be achieved if content is cached at high betweenness centrality nodes.

A few efforts [1, 2] study the coordination of caching policies in networks of caches and show the benefit of collaboration in terms of offloading the wide area network. However, this prior work assumes that all caches are accessible by all users, whereas we focus on a scenario in which caches are locally accessed by consumers and distributed throughout the network.

Cache replacement policies have been studied for a long time. For instance, Fagin [24] and Che [25, 26] have considered the LRU replacement policy which evicts the Least Recently Used. Others [27] have studied FIFO replacement policy, the performance of interconnected caches [28], or TTL policies [29], or other multi-level cache replacement policies [30].
In graph theory, centrality values indicate the importance of the nodes. A cache at the edge of the network is not necessarily closer to end-users requesting the content, whereas our use of centrality attempts to capture the relationship between a set of users, a set of content and a set of caches.

We argue that the network connectivity of caches relates to the content only partially, and there is a clear need to consider the content reachability by consumers in the network. Our review of prior work reveals that this has not been addressed by the vast majority of previous work. The work by Khan et al. [32] is the one exception. They considered content centrality in the context of fog networking, but this requirement has no impact on the content-based centrality we describe subsequently.

Each content object has a popularity. More specifically, we define by $p_x$ the probability that a node request content object $x$, where $\sum x p_x = 1$. We assume that this probability distribution is constant over the considered time period. Finally, we assume that the caching nodes are aware of the distribution $p_x$, either because it is periodically provided by the server or by using some empirical estimate from the interests that the caching node observes.

### 3.2 A Network-oriented Information-centric Centrality Metric

In graph theory, centrality values indicate the importance that nodes hold in a graph. They are typically used for identifying critical nodes in networks. There are several typical centrality metrics, including the node degree, the closeness centrality (namely, the average length of the shortest path between the node and all other nodes in the graph), the betweenness centrality (the number of times a node acts as a bridge along the shortest path between two other nodes), eccentricity (the inverse of the distance to the furthest node), and many others.

However, the ubiquitous use of caches in an ICN shifts the importance of the nodes. A cache at the edge of the network has low centrality by most common centrality measures, but the existence of a communication link between nodes $j$ and $k$.

Network connectivity can be dynamic as nodes move around and connections vary with time. We assume that during a particular time period the connecting graph topology is relatively stable and the nodes will stay connected for a period of minutes or hours, depending on the application.

The nodes in the graph may cache content by providing a capacity $C_c(v)$ of cache storage at node $v \in V$. This cache capacity could be equal to 0 for nodes who cannot or do not wish to provide storage.

We assume an ICN in which nodes forward requests (or interests) for specific content objects along the the shortest paths to the content objects. If there are several shortest paths, then the network splits the flows equally among the shortest paths.

We define the set of known content objects as $X = \{x_1, \ldots, x_N\}$ for a catalog of $N$ pieces of content, where $x_i$ is an indivisible content object or chunk in the network. Without loss of generality, we consider individual content chunks $x \in X$. In practice, larger size content can be composed of several such content chunks.

We assume that at least one node contains a content object at all time, and we denote it as the origin server node. This is a practical requirement to ensure the content is served, but this requirement has no impact on the content-based centrality we describe subsequently.

Each content object has a popularity. More specifically, we define by $p_x$ the probability that a node request content object $x$, where $\sum x p_x = 1$. We assume that this probability distribution is constant over the considered time period. Finally, we assume that the caching nodes are aware of the distribution $p_x$, either because it is periodically provided by the server or by using some empirical estimate from the interests that the caching node observes.

### Table 1: List of Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V = {v}$</td>
<td>Set of $n$ nodes</td>
</tr>
<tr>
<td>$E^v$</td>
<td>Set of edges between nodes</td>
</tr>
<tr>
<td>$X = {x}$</td>
<td>Set of $N$ content objects</td>
</tr>
<tr>
<td>$C_c(v)$</td>
<td>Cache at node $v$</td>
</tr>
<tr>
<td>$c_{w/l}$</td>
<td>Cost of content access from server/local link</td>
</tr>
<tr>
<td>$\rho_m$</td>
<td>Miss probability</td>
</tr>
<tr>
<td>$C/S$</td>
<td>Common/different content in cache</td>
</tr>
<tr>
<td>$p_x$</td>
<td>Popularity of content object $x$</td>
</tr>
<tr>
<td>$cp$</td>
<td>Cache permutations</td>
</tr>
<tr>
<td>$d(u, x)$</td>
<td>Distance from user $u$ to content $x$</td>
</tr>
<tr>
<td>$\sigma(u, x)$</td>
<td>no. of shortest paths b/w user $u$ and content $x$</td>
</tr>
<tr>
<td>$\sigma_c(u, x)$</td>
<td>no. of shortest paths b/w user $u$ and content $x$ passing through node $v$</td>
</tr>
</tbody>
</table>

Ren et al. [31] took a step in the direction of this paper by considering the topological relationship between the content, the cache and the user. The proposed MAGIC algorithm replaces content in the cache if the next content offers a larger gain. However, this attempt considers only the latest request from a single user, whereas our use of centrality attempts to capture the relationship between a set of users, a set of content and a set of caches.

We assume that a set of $n$ nodes or mobile devices are connected to a proximity network through local, ad hoc connections.

The connectivity between nodes is modeled by a graph $G(V, E^v)$ where $V = \{v_1, \ldots, v_n\}$ is the set of nodes and $E^v = \{e_{jk} | v_j, v_k \in V, j \neq k\}$ is the set of edges $e_{jk}$ modeling the existence of a communication link between nodes $j$ and $k$.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V = {v}$</td>
<td>Set of $n$ nodes</td>
</tr>
<tr>
<td>$E^v$</td>
<td>Set of edges between nodes</td>
</tr>
<tr>
<td>$X = {x}$</td>
<td>Set of $N$ content objects</td>
</tr>
<tr>
<td>$C_c(v)$</td>
<td>Cache at node $v$</td>
</tr>
<tr>
<td>$c_{w/l}$</td>
<td>Cost of content access from server/local link</td>
</tr>
<tr>
<td>$\rho_m$</td>
<td>Miss probability</td>
</tr>
<tr>
<td>$C/S$</td>
<td>Common/different content in cache</td>
</tr>
<tr>
<td>$p_x$</td>
<td>Popularity of content object $x$</td>
</tr>
<tr>
<td>$cp$</td>
<td>Cache permutations</td>
</tr>
<tr>
<td>$d(u, x)$</td>
<td>Distance from user $u$ to content $x$</td>
</tr>
<tr>
<td>$\sigma(u, x)$</td>
<td>no. of shortest paths b/w user $u$ and content $x$</td>
</tr>
<tr>
<td>$\sigma_c(u, x)$</td>
<td>no. of shortest paths b/w user $u$ and content $x$ passing through node $v$</td>
</tr>
</tbody>
</table>
is actually critical in reaching and storing the content that is requested by consumers. If the probability of a hit in the cache is, say 60%, then even if an upstream node fails in the network, the network will still satisfy 60% of the consumers’ requests. It is clear that new centrality metrics are needed for ICNs.

We propose the Network-oriented Information-centric Centrality for Efficiency (NICE) metric. For closeness, NICE is defined as the inverse of the sum of the distance from the user to the content, that is:

\[ NICE_c = \left( \Sigma_{u,x} d(u,x) \right)^{-1} \tag{1} \]

where \( u \) is the set of users, \( x \) is the set of content, and \( d(u,x) \) is the distance on the shortest path from \( u \) to \( x \). Figure 1 shows an example, which we detail in the next section.

For betweenness centrality, NICE is defined as the sum of the ratio of the number of shortest paths from all users to all content objects that passes through the node to the total number of shortest paths between all the (user, content) pairs weighted by the popularity of each content object.

Formally, for probability distribution \( p_x \) for content \( x \) and if \( \sigma_x(u,x) \) is the number of shortest paths from user \( u \) to content \( x \) going through node \( v \) and \( \sigma(u,x) \) the total number of shortest paths between \( u \) and \( x \), then:

\[ NICE_b(v) = \Sigma_{u,x} p_x \frac{\sigma_x(u,x)}{\sigma(u,x)} \tag{2} \]

We can normalize this to a measure between 0 and 1 if needed. Figure 2 shows an example, which we detail in the next section.

### 3.3 Example of NICE

#### 3.3.1 Closeness. Figure 1 shows a simple network where users connect through nodes \( v_1, v_2 \) and \( v_3 \) to access the content. Assume that each cache can hold one piece of content, and that we have pieces of content \( x_1, x_2, x_3, \ldots \) with probability \( p_1 > p_2 > p_3 > \ldots \). Further, assume that the distance of a link between the \( v_i \)'s is \( c_l \) (for some local link cost) and the distance between \( v_4 \) and the server is \( c_w \) (for some wide area link cost). In practice, \( c_w >> c_l \).

If we assume each cache works independently, \( v_1, v_2 \) and \( v_3 \) all cache \( x_1 \). This is the Least Frequently Used (LFU) cache eviction policy that would keep the content most frequently requested. Least Recently Used (LRU) similarly evict content that has not been requested the longest, and typically keeps the most frequently requested content as well. Obviously this is a poor content placement as a significant fraction \((1-p_l)\) of the traffic travels the greater distance \( c_w \).

A much better content placement is to cache all three for \( x_1, x_2 \) and \( x_3 \) in the local network. Define by \( p_m \) the miss probability \( 1 - (p_1 + p_2 + p_3) \).

If the content is placed as \( x_1 \) in \( v_1, x_2 \) in \( v_2 \) and \( x_3 \) in \( v_3 \), then we can compute the distance from \( v_4 \) to the content, namely \( p_m(c_l + c_w) + (2p_2 + p_3)c_l \). The distance from \( v_2 \) to the content is \( p_m(c_l + c_w) + (2p_1 + p_3)c_l \); and the distance from \( v_3 \) to the content is \( p_m(c_l + c_w) + (2p_1 + p_2)c_l \). The total distance is \( 3p_m(c_l + c_w) + (4p_1 + 3p_2 + 3p_3)c_l \).

Since \( p_1 > p_2 > p_3 \), we see it is beneficial to put the content \( x_1 \) in \( v_2, x_2 \) in \( v_3 \) and \( x_3 \) in \( v_1 \). In this configuration, the distance to the content becomes \( p_m(c_l + c_w) + (2p_1 + p_2)c_l \) for \( v_1, p_m(c_l + c_w) + (p_1 + 2p_3)c_l \) for \( v_2 \) and \( p_m(c_l + c_w) + (p_1 + 2p_2)c_l \) for \( v_3 \) for a total of \( 3p_m(c_l + c_w) + (3p_1 + 3p_2 + 4p_3)c_l \).
Since \( p_1 > p_2 > p_3 \), the latest placement puts the content closer to the users. The distance to the content is the measure of centrality. Nodes \( v_2 \) and \( v_4 \) are better connected to the content than \( v_1 \) due to the extra link joining them. They have higher content centrality. Therefore it makes sense to place content such that the content-based centrality is increased.

On this simple topology, the expression of the distance to the content requires computing paths from all users to all contents.

### 3.3.2 Betweenness

We now consider betweenness NICE. Figure 2 shows a graph, where nodes \( S \) is the server, and holds a copy of all the content, nodes \( v_1, v_5 \) and \( v_3 \) are cache nodes, the users are nodes \( a, (b) \) and \( (c) \) and the other nodes \( v_2, v_4, v_6 \) act as relays.

Rather than identifying exactly which content is in the cache, for simplicity we have assumed on this figure that the content is distributed in blocks, where the capacity \( C_c \) is divided into \( C \) content objects that are the same at all caching nodes, and \( S \) content objects that are unique to that cache. We assume, for ease of explanation only, that all content is equally likely.

Figure 2 also indicates the NICE value of each node, and the degree centrality as well. The (non-content-based) Betweenness centrality of this graph has little meaning, underlying the need for ICN-specific centrality measure. Computing the betweenness centrality between all the nodes does not capture that only \((a), (b)\) or \((c)\) issue requests, nor that only \( v_1, v_5, v_6 \) and \( S \) can respond to these interests. For this reason, we do not include this metric on this figure.

To compute the NICE value, let us consider node \( v_6 \) on this figure. The user \((a)\) will send traffic to \( v_6 \) only for the content that is specific to \( v_6 \). The common content \( C \) will be served by \( v_1 \) for user \((a)\). For this \( v_6 \)-specific content, half of the requests from \((a)\) will go to \( v_6 \), and half will go to \( S \), as both are four hops away from \((a)\) and therefore both are on the shortest path to this content. Therefore the contribution from \((a)\) is \( S/2 \), as all \( S \) content are equally likely.

For user \((b)\), there are three paths to the content that is \( v_6 \)-specific: \( v_7-v_5-v_3-S \), or \( v_7-v_5-v_1-v_6 \) or \( v_7-v_5-v_5-v_6 \). One of these three paths end at \( v_6 \), therefore \((b)\)'s contribution is \( 2S/3 \). Finally, user \((c)\) sends requests for the content that is \( v_1 \)-specific through either the 3-hop path to \( v_1 \) or the two 3-hop paths to \( S \), including one through \( v_6 \) for a total of \( S/3 \). All the \( v_6 \)-specific requests from \((c)\) end up at \( v_6 \) for a total of \( S \). Half of the common content \( C \) from \((c)\) ends up at \( v_6 \), the other half goes to \( v_5 \), adding \( C/2 \). Finally, the rest of the content requests go to the server \( S \) and half of these follow a path through \( v_3 \) (the other half goes through \( v_5 \)), adding \((N-C-3S)/2 \). Summing all these contributions yield \( S + N/2 \).

The other nodes’ content-based centrality can be computed in an identical manner, by identifying the contribution to each of the other contents from each user. It should be obvious from this simplified example that computing content-based centrality may become complicated, especially once we take into account the content popularity and an increasing number of users. In the previous example, it is straightforward to take into account the popularity, by identifying the aggregated probability that a piece of content is in \( C \) or is specific to one of the \( v_j \). However, in practical systems, this gets unwieldy.

We address next how to efficiently compute the NICE value at a node. While we have presented of NICE for two centrality metrics, we now focus exclusively on betweenness centrality as it is a linear expression and therefore easier to work with in practice.

### 3.4 Scalable Computation of Content-based Centrality

Computing the content-based centrality from Equation 2 may get extremely difficult, since the catalog \( X \) may be very large, and it requires knowing the content placement as well as the probability distribution \( p_x \) as well.

For this, we propose performing an offline computation for combination of caches first. Note that this scales with the number of caches, and not with the number of content, dramatically reducing the complexity.

More formally, denote by \( \{c_1, \ldots, c_m\} \) the set of \( m \) nodes \( v_i \in V \) such that \( C_c(v_i) > 0 \). These are the caching nodes in the network.

A piece of content can be in any \( 0 \leq k \leq m \) of these nodes. There are \( 2^m \) potential combinations of caches a content can belong to. Denote by \( cp \) (for cache permutation) an element of \( 2^m \), that is a set of anywhere between 0 and \( m \) caches.

For each of these combinations, we compute once the NICE value of a single item of popularity 1 located at the caches in this combination \( cp \), and no other item anywhere else. That is we consider that content \( x \) is the only content that is requested and is placed at the caches \( c_i \in cp \). We can compute the NICE value for \( x \) and \( cp \) at node \( v \).

We do this for all the combinations and store the value for each \( cp \in 2^m \). Namely, we compute \( NICE_{cp}(v) \) for all such \( cp \) cache combination and store the outcome in a table at each cache.

In practice, adding more copies of the same item in more caches has a decreasing marginal utility, and the network operator may set a limit on the number of copies for any piece of content. Therefore we do not have to compute the NICE value for every \( cp \in 2^m \); we can start by computing \( NICE(v) \) for each node \( v \) with \( cp \) of cardinality 1, 2, and keep...
increasing until we reach the limit set by the operator. We can then stop the process and start with another node w.

While this process is still relatively complex, it scales with the number of cache combinations, and has to be done offline and only once.

To compute the NICE value of a specific content placement, we only need to look up the cache permutation \( cp(x) \) of content \( x \). A content object \( x \) will belong at any point of time to one cache set \( cp(x) \) and the NICE value can therefore be computed by:

\[
NICE(v) = \sum_{x \in X} p_x NICE_{cp(x)}(v)
\]

where \( NICE_{cp(x)}(v) \) was pre-computed ahead of time.

4 CONTENT REPLACEMENT ALGORITHM IN THE CACHE

4.1 Algorithm

Equipped with the tool of content-based centrality and a method to compute it efficiently, we now use it to optimize the cache replacement policy in ICN and network of caches.

Algorithm 1 depicts our proposed cache replacement policy. The intuition of the algorithm is simple. If adding a new content into the cache \( c_j \) increases \( NICE(c_j) \), then the operation is carried on. Otherwise, the content is not added.

The system is initially empty of any content. Then, when content arrives at the cache and the cache is not full, it is added. When the cache is full, the distributed content placement is optimized at each caching node using Algorithm 1.

We denote by \( NICE_x(v) \) the contribution of \( x \) to \( NICE(v) \), namely \( NICE_x(v) = p_y NICE_{cp(x)}(v) \).

The algorithm generates the replacement policy in the cache, while attempting to always increase the centrality of the cache. Because content centrality increases, the content always gets more reachable.

In this algorithm, at node \( v \), the only computation that is required is the difference between \( p_y NICE_{cp(y)}(v) \) where the content placement \( cp(y) \) assumes \( y \) is cached at \( c_j \), and \( p_z NICE_{cp(z)}(v) \). Since \( NICE_{cp} \) is computed ahead of time, the only operation is to know \( cp(y) \) and \( cp(z) \) as well as \( p_y \) and \( p_z \).

The values \( cp(y) \) and \( cp(z) \) should be provided by the protocol to update the routing tables, especially since this update is required to take advantage of the new objects in the cache. \( p_y \) and \( p_z \) need to be either periodically provided by the server, or estimated by empirically monitoring the rate of interests for \( y \) and \( z \).

4.2 Convergence

Theorem: Algorithm 1 converges to a stable content placement policy.

The content placement algorithm only increases the NICE value at each cache for each content inclusion/eviction decision. Hence, it suffices to show that a finite number of increases to NICE values occur at any node.

We have a finite number of pieces of content, and a finite number of permutations; therefore, the increment to a NICE value is bounded below by the smallest change in NICE by which it may occur by either: (a) moving one piece of content from one cache to another, (b) adding a piece of content to a cache, or (c) removing a piece of content from a cache. Since the increment is bounded below, Algorithm 1 will converge in a finite number of decisions.

5 NUMERICAL EVALUATION

We evaluate NICE using the named-data networking module of the NS-3 simulation platform (ndnSIM [33]). Three different topologies from a large scale realistic trace of 2,986 nodes in Köln, Germany is used in order to validate the efficiency and scalability of NICE in a real edge caching environment. The Köln 6 × 6 km² city center is divided into 36 neighborhoods comprising 25 nodes each, thus, we consider a large as up to 900 caching nodes out of the 2,986 nodes. The topologies are snapshots of the network connectivity at time \( t = 0 \), \( t = 30 \) and \( t = 60 \) minutes, respectively.

5.1 Simulation Scenario

The simulation scenario implements consumer nodes which generate up to \( 10^5 \) unique content requests following a Zipf distribution (with varying Zipf parameter 0.5, 0.75, 1, 1.5). We vary the cache size from 1, to 10 and to 100 MB. We run
Towards A Scalable Information-Centric Approach to Cache Management ACM ICN'18, September 2018, Boston, MA, USA

Figure 3: Hit rate comparison by varying cache size (Zipf parameter=1) for centrality-based NICE, Degree, Closeness, Betweenness, Eigenvector, non-centrality based or social-unaware (LRU) and non-collaborative (LRU-Individual) caching.

Figure 4: Hit rate comparison by varying Zipf parameter (cache size=100 MB) for centrality-based (NICE, Degree, Closeness, Betweenness, Eigenvector), non-centrality based or social-unaware (LRU) and non-collaborative (LRU-Individual) caching.

We implement the cache replacement policy where the node caches the content that maximizes its respective NICE (considered here only to be betweenness NICE). Besides NICE, we implement three different cache replacement approaches for comparison:

- **Centrality-based**: We cache popular content at high Degree, Closeness, Betweenness and Eigenvector centrality nodes. For each of these centrality metric, we rank

Each set of parameters for a total of 100 times, each with a new random seed to generate consumer interests.

Any provider node already caching the content responds to the consumers’ interests. Intermediate nodes with homogeneous buffer size to perform in-network caching. 30% of the nodes are consumers, 30% are providers/caching nodes and the remaining nodes act only as relay nodes in the Information-centric Network.
Figure 5: Hop count comparison by varying cache size (Zipf parameter=1) for centrality-based (NICE, Degree, Closeness, Betweenness, Eigenvector), non-centrality based or social-unaware (LRU) and non-collaborative (LRU-Individual) caching.

Figure 6: Hop count comparison by varying Zipf parameter (cache=100MB) for centrality-based (NICE, Degree, Closeness, Betweenness, Eigenvector), non-centrality based or social-unaware (LRU) and non-collaborative (LRU-Individual) caching.

- The nodes by centrality, and populate the cache by placing the most popular content at the highest centrality node, the next most popular content at the next highest centrality node, and so on until all caches are full.
- Non-centrality based: We implement a collaborative approach where all the nodes in the network pool their cache space to form one large LRU buffer. This caches the most popular content in the network, but without topological considerations.
- Non-collaborative based approach: (LRU-Individual) Each node implements an LRU policy independently of the other nodes. This is a myopic greedy local policy, however it is commonly considered in Information-centric Networks.
Towards A Scalable Information-Centric Approach to Cache Management ACM ICN’18, September 2018, Boston, MA, USA

The following performance metrics are used to the cache replacement approaches:

- **Cache Hits**: It is the average number of content responses from the caching nodes, calculated as the ratio of the number of data served by the cache to the number of received interests by the nodes.
- **Distance to content (Hop count)**: It is computed as the distance in terms of average number of hops traversed by the content en route to the consumer. A lower number of hops traversed by the content high reflects better content reachability.
- **Delay in content retrieval**: The overall delay in retrieving all content requests generated by nodes.

### 5.2 Simulation Results

#### 5.2.1 Cache Hits

We observed the cache hit rate for each scheme: (i) NICE, Degree, Closeness, Betweenness and Eigenvector centrality, (ii) non-centrality based (LRU), and (iii) LRU-Individual where individual nodes cache indifferently, i.e. no collaboration on cache replacement. In Figure 3 we compare the hit rate of our approach for three topologies from the Köln trace and three cache sizes (1MB, 10MB and 100MB). Figures 3a, 3b and 3c show results for each topology. The first observation is that the average cache hit rates achieved differ with respect to topology and cache size. Furthermore, we trivially observe that for all topologies there is an increase in the cache hit rate with increasing cache size.

Further, we see that NICE achieves a high cache hit rate when compared to all other approaches. It resulted in a maximum of 83% hit rate for Topology 1, 60% for Topology 2 and 70% for Topology 3.

Similarly, Figure 4 show the hit rate comparison for each topology by varying the Zipf parameter. Here as well, NICE outperform the other schemes where there is a substantial difference (up to 2x) for low values of Zipf parameters. This is because other schemes fail to cache the maximum number of popular content for low Zipf parameter values. We also observe that all the centrality schemes resulted in substantially higher hit rate than the case LRU-Individual, which achieves the lowest average hit rate (below 50% overall) thus confirming the well-known benefit of collaborative caching. However, the caching decision in our case is fully distributed: each nodes assess the benefit on content centrality locally. NICE captures locally the impact on the overall network.

The comparative analysis of cache hit rate on three different topologies, three caches size and four variants of Zipf parameter reveals that NICE achieves a better hit rate and in a scalable manner.

#### 5.2.2 Distance to content (Hop Count)

We also compute the average distance to content from the consumers. Figures 5 and 6 show the average number of hops the content traverse from the caching node which responds to the interest, to the consumer.

Figure 5 compares the hop count for each topology by varying the cache size. We first observe that, NICE results in the least number of hops overall (around 2 hops), where the case of LRU (both with and without collaboration) yield highest distance to content. Other schemes achieve similar distance, where we observe that in general, centrality-based cache replacement results in better performance.

Note that other centrality measures, such as closeness or eccentricity, are specifically tuned to minimize the distance. However, the content-based betweenness centrality seems to perform well in this regard.

Similarly, Figure 6 compares the average hop count for each scheme by varying the Zipf parameter. We observe a variation in the hop counts between different centralities, however we observe that most of the schemes are successful in providing content within two hops, though, NICE achieves relatively better results, i.e. within a distance of 2 hops, for all variations of Zipf parameter. However, we see that LRU (both with and without collaboration) fails to provide content within 2 hops for most of the cases.

The hop count analysis overall suggests that centrality based cache replacement, and NICE in particular, provides an efficient and scalable approach to enhance cache performance in Information-Centric Networks.

#### 5.2.3 Delay in content retrieval

To validate the time efficiency of the proposed content replacement approach, NICE, we compute the average delay in seconds for the content retrieval using each scheme. Figure 7 shows the average delay observed using each topology, for a cache size of 100MB and
Zipf parameter = 1. We observe that NICE is able to achieve the least delay among the compared schemes. The topology 2 yielded high delay values (around 0.1s to 0.12s), thought, despite the high delay values as the impact of topology overall NICE outperformed the other compared schemes. Thus, we can infer that NICE is an efficient content replacement approach in ICN with respect to delay.

6 CONCLUSIONS AND FUTURE DIRECTIONS

We observed that Information-Centric networks require a new approach for cache management, as well as new graph theoretical measures. The current definitions of centrality are inadequate when dealing with connecting users to content. These measures are agnostic to the opportunity for the network to keep content in the caches. Therefore well-connected users in the traditional sense of centrality may be poorly connected in the sense of content-based centrality, if what they are looking for is not locally available.

We described NICE, a content-based centrality, and gave examples based both upon closedness and betweenness centrality. We showed a more scalable method to compute such centrality. We applied NICE to a content replacement policies in the caches. Our simple NICE-based cache replacement policy evicts content whenever the insertion of a new content object improves the content-based centrality of the node. This cache replacement policy performs extremely well in a set of simulations against LRU and graph centrality based policies. In our evaluations, we have shown that, while making a local, distributed decision to either keep or evict content in the cache, the NICE-based policy distributes the content in the network so that its average distance and delay to the users is lower and the cache hit rate is higher than the other benchmarks.

Future work is to evaluate other forms of content-based centrality, i.e. based upon closeness centrality and eccentricity along a practical implementation of NICE over NDN.

REFERENCES

Towards A Scalable Information-Centric Approach to Cache Management


