LoCHiP: A Distributed Collaborative Cache Management Scheme at the Network Edge

Junaid Ahmed Khan, Cedric Westphal, J. Garcia-Luna-Aceves, Yacine Ghamri-Doudane

LoCHiP: A Distributed Collaborative Cache Management Scheme at the Network Edge

Junaid Ahmed Khan*, Cedric Westphal†, J.J. Garcia-Luna-Aceves‡, and Yacine Ghamri-Doudane§
*New York University, NY, USA
†Futurewei Technologies, Santa Clara, CA, USA
‡University of California, Santa Cruz, CA, USA
§L3i Lab, University of La Rochelle, France
junaid.khan@nyu.edu, cwestphal@gmail.com, jj@soe.ucsc.edu, yacine.ghamri@univ-lr.fr

***Abstract*— Using local caches is becoming a necessity to alleviate bandwidth pressure on cellular links, and a number of caching approaches advocate caching popular content at nodes with high centrality, which quantifies how well connected nodes are. These approaches have been shown to outperform caching policies unrelated to node connectivity. However, caching content at highly connected nodes places poorly connected nodes with low centrality at a disadvantage: in addition to their poor connectivity, popular content is placed far from them at the more central nodes. We propose reversing the way in which node connectivity is used for the placement of content in caching networks, and introduce a Low-Centrality High-Popularity (LoCHiP) caching algorithm that populates poorly connected nodes with popular content. We conduct a thorough evaluation of LoCHiP against other centrality-based caching policies and traditional caching methods using hit rate, and hop-count to content as performance metrics. The results show that LoCHiP outperforms significantly the other methods.**

***Index Terms*—Content Centric Networking, Content Caching, Content Offload, Centrality.**

I. Introduction

Facing an exponentially growing demand [1, 2], cellular network operators have been trying to offload traffic to other networks as aggressively as possible [3]. An effective method to alleviate bandwidth pressure on the cellular links consists of using local caches at users' nodes or at the access points of small cell base stations or WiFi networks, and providing methods and incentives for users to draw traffic from local caches as much as possible [4] [5].

Traditional caching policies consist of keeping popular or recent content, using First-In First-Out (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU) [6][7][8]. These are non-managed caching policies, where an individual cache of a caching network receives new content periodically, and upon seeing this new content, decides whether to keep it or not, and if needed, which other piece of content to discard to make room for this. In typical Content Delivery Networks (CDNs), the operator of the network of caches assigns some content to caches according to some optimization principles.

It makes sense to take into account the topology around caches for content placement, and in particular the connectedness of caching nodes. Centrality is a concept used to identify the most connected nodes in a graph. A high centrality score reflects a high topological connectivity for a node in the network. In the past, the concept of centrality has been used with node connectivity having social or physical meanings. For example, it has been used in the analysis of social networks to find influential users in a social network [9]. Centrality has also been used with physical connectivity in mind, and a number of approaches [10, 11] advocate caching popular content at nodes that have high centrality, i.e., that are well connected. Section II discusses relevant prior work, and it is important to note that prior caching policies that take advantage of the connectivity of caching nodes place popular content at those nodes that have high centrality, which we call High Centrality High Popularity (HiCHiP) caching policies.

Contrary to the conventional wisdom regarding caching policies that consider node connectivity, we believe HiCHiP caching policies are ill-suited for traffic offload: It is a case of *the rich getting richer*. In an area with many densely-connected nodes, base stations, access points, a richly connected node will have access to plenty of content, and will be able to pick from the most popular. A poorly connected node at the edge of this area on the other hand will have few caches nearby from which to fetch data, and these caches at the edge will have low centrality. If caching is based on HiCHiP, poorly connected nodes will hold low popularity content as well.

The rationale for HiCHiP caching policies in the past has been the prevailing view that *a high-centrality node is a natural candidate to* **provide the content to** *other nodes*. However, when centrality refers to physical connectivity, cost and system efficiency dictate that there can only be a few high-centrality nodes in a network, and hence an HiCHiP caching policy results in the most popular content being concentrated at a few nodes, and the system may not be able to deliver much of this content to requesting nodes because natural bottlenecks are created.

In this paper, we argue that a far better way to take node connectivity into account in a caching policy consists of reversing the meaning of connectivity insofar as content delivery is concerned. Specifically, *a high-centrality node is a natural candidate to* **obtain the content from** *other nodes*. According to this view, we propose a new network cache management policy, namely, a Low-Centrality High-Popularity (LoCHiP) policy for content placement, so that a node that has poor connectivity at least will have popular content nearby; while a well connected node may find it from one of its many neighbors. There is no performance loss if a high-centrality node does not have the content in its cache: it can find it nearby. On the other hand, a poorly-connected node that does not have some content in its cache has few opportunities to get it otherwise. LoCHiP based cache management policy ensures that the content of a poorly connected cache is as useful as possible.

The main contribution of this paper consists of formalizing this intuition by validating that LoCHiP indeed performs better than HiCHiP or that non-managed policies such as LRU.

The paper is organized as follows: Section II discusses the related work; Section III describes LoCHiP in details. Section IV discusses a distributed algorithm for collaborative content placement based on LoCHiP where low-centrality nodes cache the most popular content, and high-centrality nodes cache the most popular content that is not directly available in their connected neighborhood (as in any caching policy, unpopular content is not cached). Section V presents the results of extensive simulations that validate LoCHiP on both static and dynamic realistic topologies comprising $2,986$ nodes. Section VI concludes our paper with some insights into future directions.

II. Related Work

Network centrality schemes such as Degree, Closeness, Betweenness and Eigenvector centrality are well known tools to identify important nodes in networks. Social networks analysis exploit centrality to find influential information hubs for publishing/spreading information or advertisements.

Pantazoupoulos et al. [12] defined a variant of typical betweenness centrality named as conditional betweenness centrality (CBC) to place content. The authors considered a subset of nodes with high CBC for content caching. However, no solution is provided on which content should be placed at each node. Chai et al. [10] proposed centrality-based caching algorithm by exploiting the concept of betweenness centrality to improve the caching gain. Wang et al. [13] solve an optimization problem to find where to place the content in order to minimize the cost of delivery. A solution is provide to address how much cache should be allocated at each node. By contrast, our target is to address which nodes should cache which content in the network.

Rossi et al. [14] proposed to size the cache of different nodes as a function of different centrality metrics such as degree, betweenness, closeness, graph, and eccentricity. They show little benefit of using centrality to size caches, which supports our claim that high centrality should not be where to place the content first. In a similar context, Yufei et al. [15] use "control nodes" to cache content based on betweenness centrality. Nguyen and Nakazato [16] proposed a betweenness-centrality-based network coder placement for peer-to-peer (P2P) content distribution. However, they do not consider content placement; they use random linear network coding at every coding node to assist content delivery. Li et al.[5] use game theory for caching popular videos at small cell base stations (SBSs) while Mehr et al. [17] suggest to prefetch video content in caches to improve Quality of Experience for users at the network edge. Mangili et al. [18] proposed a game theoretic approach in Information Centric Networks (ICN) to stimulate wireless access point owners to jointly lease their unused bandwidth and storage space to a content provider under partial coverage constraints. Both efforts targeted a pricing model instead of providing an efficient content-placement solution.

Distributed caching in ICN is targeted by different studies [19, 20, 21]. Wang et al. [22] address the distribution of the cache capacity across routers under a constrained total storage budget for the network. They found that network topology and content popularity are two important factors that affect where exactly should cache capacity be placed. Yu et al. [23] looked at pushing content to the edge to anticipate network congestion, while Azimdoost et al. [24] computed the capacity of an ad-hoc network of caches.

III. Centrality-based Content Placement

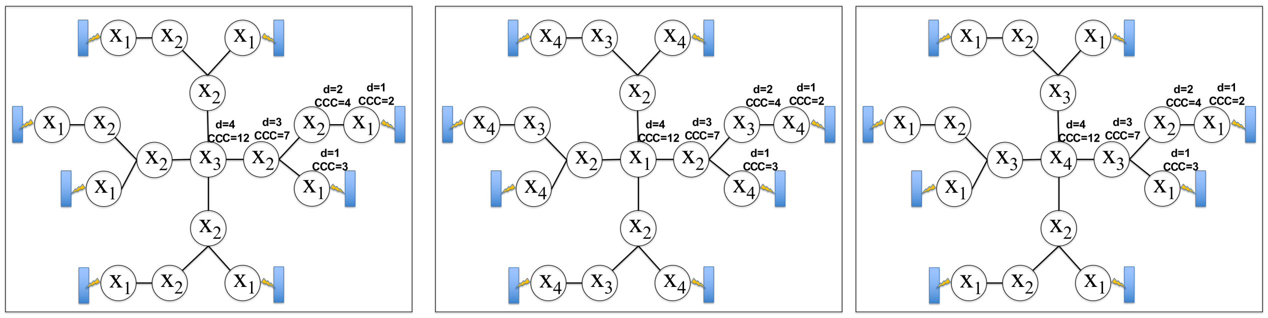
A. Motivating Example

We consider the illustrative example shown on Figure 1. We define the known content to be $\mathbb{X} = \{x_j, j = 1, \ldots, N\}$ where x_j is an indivisible content chunk in the network, and N is the (potentially very large) number of pieces of content. In the remaining of the paper, with no loss of generality, we consider individual content chunk x. We assume a set of connected nodes can each cache exactly one piece of content from the content set $\mathbb{X} = \{x_1, x_2, x_3, x_4, x_5\}$[1]. Each content is requested with popularity $p(X) = \{p_1, p_2, p_3, p_4, p_5\}$ by the nodes at the edge of the network, where $p_1 > p_2 > p_3 > p_4 > p_5$. We assume that nodes can fetch traffic from at most h hops (say, $h = 3$) away, so as to meet delay constraints.

In each of the three panels, we can see the caching decision of the nodes. In (a) the content is cached greedily at each node, namely each node keeps the most popular content for which it sees requests(Figure 1a). This means the nodes near the edge will cache x_1, and since requests from x_1 will be served from there, the next node towards the center will cache x_2, etc. In (b), the most popular content is cached at the High Centrality nodes. This is the HiCHiP policy. The centrality can be either degree (noted d on Fig. 1) or the cache size in the 2-hop neighborhood (denoted as CCC for Cache Connectivity Centrality on Fig. 1) as in this example, both degree and CCC have the same order. We can verify that x_1 is stored in the middle of the graph, and x_4 at the edge. Finally, in (c) we describe the caching policy for LoCHiP (where the centrality is d or CCC). Now x_1 is cached at the edge.

For each of these scenarios, we can compute the rate of traffic request that are issued to the cellular network operator from edge users, or conversely, the rate of offload. None of the caches have space for x_5, which is the less popular tail of

[1]Each x_i may correspond to a group of content with similar popularity of total size comparable with that of a cache.



(a) LRU/Greedy (b) HiCHiP (c) LoCHiP

Figure 1: Content placement strategies

the content, so downloading x_5 will incur the same costs for all three scenarios (and therefore we ignore it).

For (a), the content x_4 with popularity p_4 is not cached locally thus it will be downloaded from the Service Provider (SP) by all users. The content x_3 with popularity p_3 is cached at the most central node, and is unreachable for users more than three hops away, thus, they will download it from the infrastructure. The total content downloaded from the infrastructure is: $p_3(x_3) * 4 + p_4(x_4) * 8 = 4(p_3 + 2p_4)$.

For (b), the most popular content x_1 at the most central node is unreachable from half of the users since it is more than 3 hops away. Thus, the total content downloaded from the SP are $p_1(x_1) * 4 = 4p_1$.

For (c), a less popular content x_4 is cached at the most central node and is downloaded from the SP by the users more than three hops away from it. In this case, the total content downloaded from the SP are $p_4(x_4) * 4 = 4p_4$.

It is easy to see that (a) incurs $4(p_3 + p_4)$ more download traffic from the infrastructure than (c); and the (b) incurs $4(p_1 - p_4)$ more download traffic. LoCHiP outperforms the other two proposals for any p_i satisfying $p_1 > p_2 > p_3 > p_4 > p_5$.

We note that with LoCHiP, the popular content *is still reachable* to the highly connected nodes. Indeed, in (c) the most central node has access to x_1, x_2, x_3, x_4 despite caching the least popular content (among those cached in the network). Our algorithm ensures that the most popular content is reachable *for all the nodes*, as we will see below.

Our example shows the benefit of LoCHiP for a generic distribution (as long as $p_1 > p_2 > p_3 > p_4$; p_5 corresponding to the non-cached tail may have a higher probability if it is aggregating multiple less popular items). For lack of space, we cannot include other illustrative topologies. However, other topologies show that LoCHiP perform at least as well as greedy or HiCHiP. This is confirmed by our simulations, where the topologies are randomly arising out of dynamic networks.

IV. LoCHiP-based Optimal Placement

We define an optimization model with the objective function is to minimize the overall cost for all users in retrieving content from the service provider:

$$L = \text{minimize}\left(\sum_{U}\sum_{X} c_{u,x} p_{u,x}\right),$$
$$s.t, \sum_{x} b_{v,x} \leq b_v, \text{ and } d(u, x_u) \leq h, \forall u$$

where the loss function L minimizes, for the set of users $\mathbb{U}$ their cost $c_{.,.}$ to retrieve all contents in the set $\mathbb{X}$. The first constraint deals with the buffer requirements for a node v to not cache content exceeding its buffer size b_v. The second constraint limits user u to retrieve the content x_u from a maximum of h hops away.

We allow nodes to mutually cache the maximum amount of content in a distributed manner. Nodes with caches in a neighborhood can self-organize and estimate the content popularity online based on the amount of user interest received for the cached content. We present below a heuristic for a set of nodes to opportunistically cache content in a neighborhood.

A. Distributed Content Placement

Algorithm 1 is used for content placement by each node. Node v first defines its h-hop neighborhood as the set $S_v \in \mathbb{V}$ and exchanges information regarding its cache size with the nodes in S_v (Lines 4 and 5). Using the cache size information in the neighborhood, it then computes and exchanges its respective centrality CCC_v (Line 9) with the nodes in S_v. Each node compares its CCC with the other nodes in their S_v and in case it has the lowest CCC, it initializes the placement by caching the most popular content x from the content list $\mathbb{X}$ in order to have access to the most popular content. In Line 7, a node that has filled up its buffer already indicates so, and is not considered as a candidate to be the lowest CCC node any longer. The next lowest CCC node can then fill up its cache by drawing on the next most popular content.

A popularity tag can be used for each content by the operator where nodes can recognize content based on its tag, i.e. most popular content tagged as popularity level 1, lesser popular as level 2 and so on. For ease of explanation, assume that the amount of content tagged with the same label is comparable to the size of one cache, except for the tail of the content that includes all the content that cannot be cached at all.

Thus, for a node v in a neighborhood S with k nodes having a lower centrality than CCC_v, v will cache content with the $(k+1)_{th}$ popularity level content in its placement policy. This content can be cached opportunistically (if the user sees content with this tag, it keeps it) or the cache can be populated with this content by the SP, say during periods of low demand. We denote by X_S the content cached in the neighborhood S and by X_v the content cached at node v.

Line 11 of Algorithm 1 ensures that the occupied buffer space b_v does not surpass the total available node buffer b_v^t while $X_S < \mathbb{X}$ allows the node to cache content with decreasing popularity order (Line 12) as long as there exist content to populate its cache until no more content are left to cache. The node updates the amount of individually occupied buffer space b_v and X_v the set (list) of content it cached (Line 14). The content availability in the neighborhood S is updated as shown in Line 15 and the set of content cached at the node X_v are added to the set of contents in the neighborhood X_S (Line 16).

In order to achieve collaborative caching at the set of nodes in the neighborhood, in the increasing order of node centrality CCC, the other caching nodes in the neighborhood S_v perform the same steps and cache content in its decreasing popularity order. The algorithm converges until either there is no more content left to cache or the corresponding nodes buffers are full as mentioned in Line 11. In case when a node buffer is full, it can exempt itself from the caching process. The nodes with still buffer remaining repeat the process only now considering the content that is not already placed in the neighborhood to add in the unfilled caches.

Algorithm 1: Content Placement Algorithm at Node v

```
 1: INPUT: $S$, $\mathbb{X}$, $P$, $h$, $b_v$, $\forall v \in \mathbb{V}$
 2: OUTPUT: $X_{S_v}$, $X_v \in \mathbb{X}$
 3: for each node $v$ do
 4:     Define its $h$-hop neighborhood $S_v \in \mathbb{V}$,
 5:     Exchange buffer size $b_v$ in $S_v$
 6:     Find $CCC_v$
 7:     if buffer $b_v$ is full then
 8:         set $CCC_v$ to $+\infty$
 9:     end if
10:     Exchange $CCC_v$, $\forall v$ in $S_v$
11:     if $CCC_v = \arg\min_{v' \in S_v}(CCC)$, $v' \neq v$ then
12:         while $(b_v \leq b_v^t)$ or $(X_S < \mathbb{X})$ do
13:             $b_v \leftarrow \arg\max_{x \in \mathbb{X}, x \notin X_S} (p_x)$
14:         end while
15:         Update $b_v, X_v$
16:         $X_S \leftarrow X_S \cup X_v$
17:     end if
18: end for
19: return $X_v, X_S$
```

B. Content Retrieval

The proposed collaborative caching scheme can be complemented by a content retrieval mechanism as described in

Algorithm 2: Content Retrieval

```
 1: INPUT: $S_v$, $CCC_v$ for $v \in S_v$, $h$
 2: for a node $v$ receiving $INTEREST(x)$ do
 3:     if $x$ in $b_v$ then
 4:         $RETURN(x)$
 5:     end if
 6:     if in $S_v$ exists $v' = \arg\max_{v' \in \mathbb{V}, v' \neq v}(CCC)$, then
 7:         $FORWARD(x, v')$
 8:     else
 9:         $FLOOD(INTEREST(x), S_v)$
10:     end if
11: end for
```

Algorithm 2. For a node v in the h-hop neighborhood S_v receiving user interest for a content x, it verifies in its buffer b_v (or Content Store - CS) whether the content is already cached and return the desired content (Lines $2-5$). In case the content is not in the node's cache, it looks for a node v' with a higher centrality CCC within its h-hop neighborhood and forwards the interests to it as show in Line $6-7$. In case of failure to find a higher CCC node, the node floods the interests in its h-hop neighborhood S_v, where the interest still remains in a local broadcast scope to a maximum of h hops.

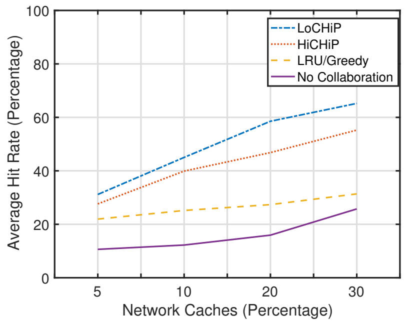
The routing tables are populated at the high CCC nodes in the h neighborhood where routing entries at such nodes pertain their ability to know where the content can be found. For instance, based on its better connectivity, a high CCC node can easily route the interest to the low CCC node caching the corresponding content. In case the content is not available in the neighborhood, the user can retrieve it directly from the SP using the costly infrastructure network as the last resort solution.

V. Numerical Evaluation

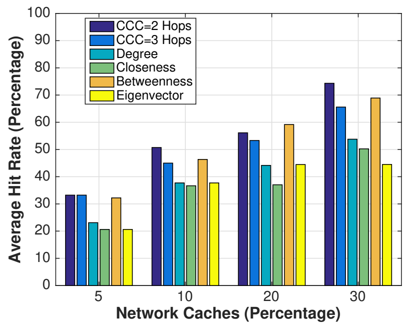
In our previous work [25], we presented analytical models to show the benefit of the proposed approach with respect to the average cost (hops) for retrieving content for an arbitrary leaf user on lattice and tree based topologies with varying Zipf skewness parameters. We theoretically evaluated the proposed LoCHiP based policy in comparison with HiCHiP and greedy cache management policy and the results suggest it to be better performing in reducing the cost of retrieving content in the network.

To further evaluate our algorithm in a network of caches, we implement a CCN-based content caching and retrieval [26] using NS-3-based simulator for the named-data networking model, ndnSIM [27] . We extract the node mobility from a realistic model of large scale dynamic mobility of nodes in Köln, Germany [28] to consider scalable caching with the existence of extreme case of mobile nodes.

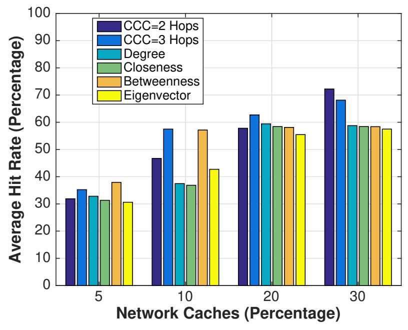
A total of $2,986$ nodes are used to validate the scalability of our caching approach. The analysis is performed by dividing an area of 6×6 km^2 into 36 uniform neighborhoods each of 1×1 km^2. Each neighborhood comprises a set of nodes, both



(a) Overall Comparison



(b) HiCHiP



(c) LoCHiP

Figure 2: Average cache hit rate compared using different centrality schemes (Static Topology)

static and mobile nodes to form a proximity neighborhood. A total duration of 1 hour of simulation is performed where the temporal characteristics of the network are captured with a granularity of 1 second. For the static topology evaluation, we consider nodes connectivity graph at three instants from the trace, i.e. at time = 0, 30 and 60 minutes, respectively.

A. *Simulation Scenario*

The simulation scenario implements an ad-hoc network with two types of nodes: (a) Consumer or user nodes generating interests with a frequency of 100 interests per second for one hour randomly for content from a catalog containing 100 unique items, comprising $10,000$ interests/sec and (b) nodes caching content as "provider" nodes in the ad-hoc network with a uniform cache size. Each simulation is repeated 10 times with random seed for consumer interests generation where results are obtained with 95% confidence intervals. The aggregate user interests follow a Zipf distribution [29] with $\psi = 1$ as the skewness parameter for the content popularity. The interests propagation is limited to maximum $h = 3$ hops in order to implement a local neighborhood scope as an upper bound on the content reachability.

For the case of CCC, we consider two variants, $h = 2$ and $h = 3$ hops as the local neighborhood limit. We compute CCC on (i) $h = 2$ hops where highest centrality CCC node is the one with the maximum amount of cache in its 2 hop neighborhood as a practical centrality exchange between neighbors, and (ii) $h = 3$ hops with the node with highest CCC is the one with the maximum cache available to it in a distance of 3 hops. Thus, in this later case, nodes farther than 3 hops are considered unreachable to the end users (consumers). The nodes's CCC is shared with other nodes within the proximity network in order to perform caching decisions locally.

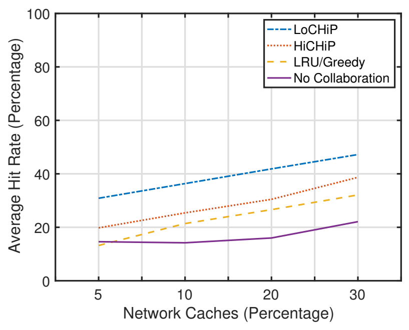
The number of caches in the network are varied from 5% to 30% since a subset of nodes can be enabled with caches in a typical network. Thus, among the $2,986$ nodes, we consider 150, 300, 600 and 900 nodes as 5%, 10%, 20% and 30% network caches respectively.

B. *Results*

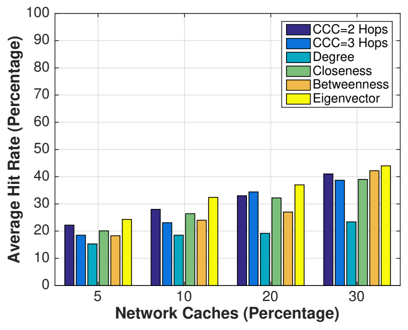
1) Hit rate: Figure 2 shows the cache hit rate computed on average of three static topologies. In Figure 2a we compare the four approaches, our proposed centrality-based LoCHiP approach, the centrality-based HiCHiP approach, the non-centrality based LRU/Greedy approach, and the non-collaborative approach. The set of nodes we consider vary from 5% to 30% caching nodes in the network. We see overall, that using LoCHiP results in high average cache hit rate with a maximum of around 66% for 30% network caches. It is followed by HiCHiP with maximum 56%, yet another centrality-based approach. On the other hand, Greedy non-centrality based caching of popular content at each node resulted in a maximum of 31% cache hit rate for similar number of network caches.

Furthermore, we evaluate both centrality-based approaches HiCHiP and LoCHiP to find out which approach yields better cache hits as shown in Figure 2b and 2c. For the case of HiCHiP, the average cache hits by node is analyzed by varying the number of nodes in the proximity network. CCC with $h = 2$ hops provided a high hit rate due to content reachability within 2 hops of consumer, while in CCC with $h = 3$ hops, the high centrality nodes are situated one hop farther compared to high centrality nodes using CCC with $h = 2$ hops. Figure 2c show that the hit-rate by all centrality schemes using LoCHiP is more than the case of HiCHiP, validating our claim. However, we see variations among different centrality schemes. This is due to the fact that most of centrality schemes identified similar nodes as the low centrality nodes, thus resulting in similar performance. We also notice that for the case of CCC, $h = 3$ hop neighborhood yields better results compared to other schemes since at most content is cached at $h = 3$ hop distance from the consumer nodes. Though for the case of 30% network cache, CCC with $h = 2$ hops dominates with around 72% cache hits as it is more likely to find content within 2 hops when there exists a high fraction of caching nodes in the neighborhood.

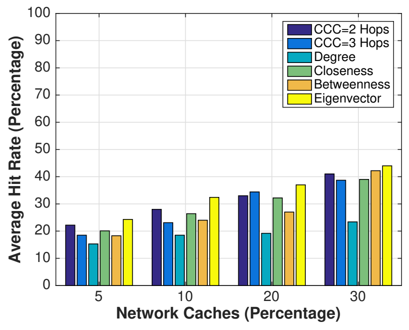
Figure 3 depicts the cache hit rate for the dynamic topology representing the existence of mobile nodes. The first observation from Figure 3a is that despite high mobility, particularly in the case of vehicles in the network, LoCHiP outperforms all other approaches. It achieves a hit rate of up to 48%, where HiCHiP, with a maximum hit rate of 40% follows. Greedy edge caching does not resulted a hit rate higher than 31%,



(a) Overall Comparison

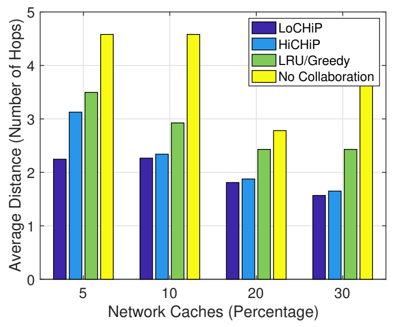


(b) HiCHiP

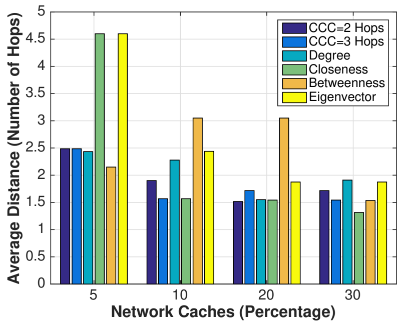


(c) LoCHiP

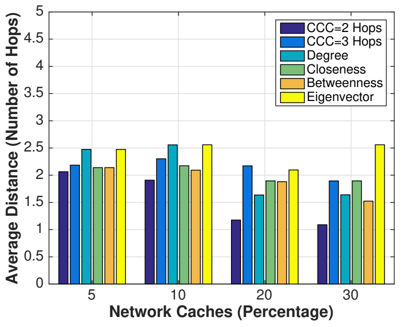
Figure 3: Average hit rate compared using different centrality scheme (Dynamic Topology)



(a) Overall Comparison



(b) HiCHiP



(c) LoCHiP

Figure 4: Average hop count compared using different centrality schemes (Static Topology)

validates our claim that even in a dynamic topology, centrality-based caching dominates where LoCHiP dominates.

On the other hand, for HiCHiP, Eigenvector centrality resulted the best performance in terms of cache hits with a maximum of 43%, still yielding a lower performance than LoCHiP. It is because LoCHiP privilege nodes with low connectivity, however, such nodes are found to be more towards network edge. Moreover, LoCHiP allows isolated nodes to cache content, thus providing access to content at nodes unreachable from the most central part of the network. At the same time, CCC=3 hops place content in 3-hop neighborhoods, thus, ensuring interests are satisfied within 3 hops from the end users.

2) Hop count: Figure 4 shows the average number of hops the content traverses from the caching node to the consumer. Figure 4a clearly shows that centrality-based LoCHiP approach outperforms the one based on HiCHiP, the non-centrality based and the non-collaborative approach. We see an average hop count of 2 for LoCHiP where HiCHiP follows with a slight improvement where the non-collaborative case results in the worst performance with an average of 4.5 hops as distance traversed, more than double of LoCHiP. Again the obvious reason for this is that LoCHiP caches content at low centrality nodes closer to consumer nodes which are rather isolated in the network.

Figures 4b and 4c compare the average hop count for centrality-based HiCHiP and LoCHiP respectively. We observe that LoCHiP achieves better results. It is evident since content is cached in 2 hop neighborhood, thus increasing the chance of finding content within 3 hops from the consumer nodes. Moreover, a decrease in hop count is noticed with increase in network caches, though a slow decrease suggests that increasing caching nodes provides less benefit and incurs cost.

VI. Conclusions and Future Work

This paper targeted the distributed cache management problem at the network edge. A centrality-based content placement policy, LoCHiP, is proposed for caching at the nodes near end-users in the network. Counter the common intuition of caching content at high centrality nodes, LoCHiP starts placing popular content at low centrality nodes since their caching resource needs to be well utilized to compensate for the poor connectivity. Additionally, low centrality nodes are often at the edge of the network, and therefore closer to the users. We evaluated LoCHiP using large scale topologies from realistic mobility trace containing $2,986$ nodes. It yielded up to 73% cache hit-rate.

Future work includes the extension of the content retrieval towards investigating the caching nodes behavior in delay sensitive dynamic networks with multiple content providers and focusing on mechanisms to avoid redundant content downloads.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020," white paper - http://goo.gl/l77haj."

[2] F. Rebecchi, M. D. De Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 580–603, 2015.

[3] "R. Gupta and N. Rastogi. LTE advanced: LIPA and SIPTO. https://www.aricent.com/pdf/aricent lipa sipto whitepaper.pdf."

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[5] J. Li, H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo, "Pricing and resource allocation via game theory for a small-cell video caching system," *IEEE Journal on Selected Areas in Communications*, 2016.

[6] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social-caching in wireless small cell networks," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on*. IEEE, 2014.

[7] E. Baştug, M. Bennis, M. Kountouris, and M. Debbah, "Cache-enabled small cell networks: Modeling and tradeoffs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, 2015.

[8] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1107–1115.

[9] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.

[10] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *International Conference on Research in Networking*. Springer, 2012, pp. 27–40.

[11] T. Le, Y. Lu, and M. Gerla, "Social caching and content retrieval in disruption tolerant networks (dtns)," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*. IEEE, 2015, pp. 905–910.

[12] P. Pantazopoulos, I. Stavrakakis, A. Passarella, and M. Conti, "Efficient social-aware content placement in opportunistic networks," *IFIP/IEEE WONS*, pp. 3–5, 2010.

[13] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2013, pp. 1–10.

[14] D. Rossi, G. Rossini *et al.*, "On sizing ccn content stores by exploiting topological information." in *INFOCOM Workshops*, 2012, pp. 280–285.

[15] C. Yufei, Z. Min, and W. Muqing, "A centralized control caching strategy based on popularity and betweenness centrality in ccn," in *Wireless Communication Systems (ISWCS), 2016 International Symposium on*. IEEE, 2016, pp. 286–291.

[16] D. Nguyen and H. Nakazato, "Centrality-based network coder placement for peer-to-peer content distribution," *International Journal of Computer Networks & Communications*, vol. 5, no. 3, p. 157, 2013.

[17] S. K. Mehr and D. Medhi, "QoE performance for DASH videos in a smart cache environment," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 388–394.

[18] M. Mangili, F. Martignon, S. Paris, and A. Capone, "Bandwidth and cache leasing in wireless information centric networks: a game theoretic study," *IEEE Transactions on Vehicular Technology*, 2016.

[19] C. Westphal, S. Lederer, D. Posch, C. Timmerer, A. Azgin, W. S. Liu, C. Mueller, A. Detti, D. Corujo, J. Wang *et al.*, "Adaptive video streaming over information-centric networking (icn)." *RFC*, vol. 7933, pp. 1–40, 2016.

[20] A. Ramakrishnan, C. Westphal, and A. Markopoulou, "An efficient delivery scheme for coded caching," in *2015 27th International Teletraffic Congress*. IEEE, 2015, pp. 46–54.

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

[22] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 95–107, 2016.

[23] Y.-T. Yu, F. Bronzino, R. Fan, C. Westphal, and M. Gerla, "Congestion-aware edge caching for adaptive video streaming in information-centric networks," in *IEEE CCNC Conference*, 2015.

[24] B. Azimdoost, C. Westphal, and H. Sadjadpour, "Fundamental limits on throughput capacity in information-centric networks," *IEEE Transactions on Communications*, 2016.

[25] J. A. Khan, C. Westphal, J. Garcia-Luna-Aceves, and Y. Ghamri-Doudane, "Reversing the meaning of node connectivity for content placement in networks of caches," in *Computing, Networking and Communications (ICNC), 2020 International Conference on*. IEEE, 2020.

[26] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[27] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of NDNsim: An open-source simulator for NDN experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.

[28] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 5, pp. 1061–1075, 2014.

[29] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *INFOCOM'99*, vol. 1. IEEE, 1999, pp. 126–134.