Study on the Theoretical Framework of Not So Cooperative Caching

Xiaoyan Hu, Jian Gong
School of Computer Science & Engineering, Southeast University, China
{xyhu, jgong}@njnet.edu.cn

Abstract

This work proposes a scheme that enables selfish nodes to cooperate in caching, here dubbed *Not So Cooperative Caching* (NSCC). We consider a network comprised of selfish nodes; each is with caching capability and an objective of reducing its own access cost by fetching data from local cache or from other caches. The challenge is to determine what objects to cache at each node so as to induce low individual node access costs, and the realistic access "price" model which allows various access "prices" of different node pairs further complicates the decision making. Using a game-theoretic approach and considering the various access "prices," NSCC seeks a global object placement in which individual node access costs are reduced as compared to that when they operate in isolation (referred to as GL) so as to incur implicit cooperation even among these selfish nodes. Our extensive experimental results demonstrate that in most cases, NSCC outperforms previous work (TSLS) which ignores the difference in access "prices," reduces individual node access costs by on average, more than 47.32% as compared to GL, and allows for a fairer treatment of nodes according to their average access "prices."

Keywords: Content caching, Selfish nodes, Game, Various access prices.

1 Introduction

This work proposes a scheme that enables selfish nodes to cooperate in caching. We consider a network comprised of selfish nodes; each is with caching capability and an objective of reducing its *own* access cost. The model assumes that the "price" of accessing an object from a node's local cache is minimal, and that from other caches are larger but smaller as compared to that from original data sources (e.g., fetching data from local cache or from a neighboring cache may reduce latency, load on potentially expensive upstream links, and so forth). The challenge is to *determine what objects to cache at each node (resulting in a global object placement)* such that these selfish nodes have incentive to join the cooperation.

This work is similar to "Cooperative Caching" [1-4], but dubbed Not So Cooperative Caching (NSCC) as each node seeks to maximize only its own benefit in terms of cost reduction rather than common welfare, which may come at the expense of others. These selfish nodes make placement decisions based on only local request patterns regardless of requests from other nodes. However, these selfish nodes can still implicitly cooperate by sharing their cached data. Then with limited storage space, each node can make its placement decisions based on the placement decisions at other nodes with an attempt to greedily minimize its own cost of serving all its requests. But as a rational and selfish entity, a node would join the cooperation if and only if its cost would be reduced as compared to that when it operates in isolation using Greedy Local strategy (GL) to cache its most popular objects, which is the rational participation constraint (i.e., mistreatment-free).

Laoutaris et al. [5] devised a mistreatment-free TSLS policy based on a game-theoretic formulation for selfish replication. A key assumption in developing TSLS is that the "prices" of accessing an object between nodes are equal which is fairly impractical. In Section 4, with an example, we show that in certain scenarios where the access "prices" of different node pairs differ, TSLS which ignores access "price" difference may even cause the cost of certain node(s) to be larger than that under GL. So this work generalizes the problem in [5] to the more practical scenarios which allow various access "prices" of different node pairs and thus each node has to distinguish copies of an object at different caches according to their access "prices" when making caching decisions. Considering the selfishness and rationality of NSCC nodes and the conflict and cooperation between them, a game-theoretic approach is applied to identify a global object placement which satisfies the rational participation constraints of all nodes (called a guaranteed object placement) so as to incur implicit cooperation even among these selfish nodes as they attempt to do better than that under GL. Note that in NSCC, once nodes commit to a specific object placement, they cannot deviate from it (i.e., no replacement is allowed) until the game is re-invoked and thus NSCC refers to object caching for a longer term. Our extensive experimental results demonstrate that in most cases, NSCC outperforms TSLS, reduces individual node access costs by on average, more than 47.32% compared to GL, and allows for a fairer treatment of nodes according to their average access "prices."

We believe NSCC is a necessary evolution of cooperative caching in the emerging world of Information

*Corresponding author: Xiaoyan Hu; E-mail: xyhu@njnet.edu.cn
DOI: 10.6138/JIT.2014.15.3.04

Centric Networking (ICN) such as TRIAD [6], DONA [7] and NDN [8]. Many ICN network architectures feature routing by name and in-network caching capability to improve performance. NSCC is well-suited to the cooperation among NDN border routers with caches at the network layer [9]. As such, existing and future applications would benefit from caching without requiring specific configurations.

2 Related Work

Cooperative caching among file/web ecaches [1-4] or among L2 caches in a multiprocessor system [10-11] seeks a "globally beneficial placement" of items to improve cache hit ratio and item access latency for the whole system (i.e., common welfare). Such caches belong to one organization and share no fundamental conflict of interests but broad common interest. They would like to agree upon a "globally beneficial placement," e.g., in [10], a L2 cache spills replaced blocks to other L2 caches to avoid future off-chip accesses and the latter host cache accepts such placement, which may result in the eviction of data at the host cache (as a subsequent spill is not allowed). However, a "globally beneficial placement" is much harder to achieve in a network with selfish caches (our NSCC case) than in a network with caches friendly to each other, as in the former, caches behave as rational entities that aim at maximizing its own benefit, which may come at the expense of others.

To the best of our knowledge, there are only a few recent works on game-theoretic aspect of cooperative caching. The work in [12] which serves as the seminal work on game-theoretic aspect of cooperative caching studies selfish cooperative caching without consideration of storage limitation. But cache-capacity limitation models an important real-world restriction and hence the following works [5][13-14] focus on the capacitated version which is left as an open direction by [12]. They all consider distributed and capacitated selfish caching and follow the simplified access "price" model introduced in [15] where nodes are equidistant (equal access "price") from one to another and a special data source holds all objects. The work in [5] devises a cooperative caching strategy (TSLS) among selfish nodes such that Nash equilibrium object placement is obtained. However, our work shows that with TSLS, in a realistic environment where different node pairs are with different access "prices", access costs of certain nodes may be even larger than that when they operate in isolation. The work in [13-14] extends the work in [5] with node churn, i.e., random changes in the set of participating nodes in the group that may occur due to "join" and "leave" events, and studies corresponding game theoretic properties. Pollatos et al. [16] slightly extend the work in [5] to the case that where special data sources for different objects are at different distances. Gopalakrishnan et al. [17] indeed extend the access "price" model into the realistic scenario which allows various access "prices" of different node pairs so as to model the more generic cases and allow a more extensive application prospect and they primarily focus on the discussion of the existence of Nash equilibrium object placements in theory, do not devise a feasible algorithm to seek an object placement that enables selfish nodes to cooperate in caching, and not to mention experimental analysis. Our work follows the realistic access price model in [17], but instead, we focus on devising an algorithm to seek a guaranteed object placement and verifying its effectiveness both in theory and with extensive simulations.

Another related work was done on the market-based resource allocation in content delivery networks [18], where the authors consider only greedy local replication strategies. There is another line of works on incentives in P2P networks. e.g., Antoniadis et al. [19], study the problem of attracting users to a P2P network and making them contribute more content. The aforementioned work and other similar ones, formulate the problem at a completely different level as compared to our work, as they focus on the number of files shared by each node, without identifying the identities of these files, whereas we focus on identifying the exact set of files that can be shared by nodes.

3 Problem Formulation

We formally define the NSCC problem as follows. As illustrated in Figure 1, we are given a set of n selfish caching nodes forming a "NSCC group," and a set of m unit-sized objects. The access pattern of node i is described by vector $r_i = \{r_{i1}, r_{i2}, \ldots, r_{ik}, \ldots, r_{im}\}$ where r_{ik} is the rate at which node i requests object k.

Each node aims to minimize its own access cost. When node i accesses an object, the cost depends on the object's location. Let $d_{i,j}$ denote the cost for node i to access an object cached at node j, $d_{i,i}$ denote the cost to fetch an object

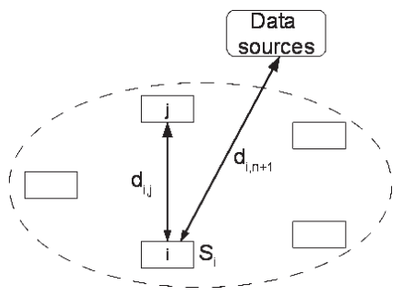


Figure 1 An NSCC Group

from its local cache and $d_{i,n+1}$ denote the cost for node i to fetch an object from original data sources. We assume $\forall i$, j, $d_{i,i} < d_{i,j} = d_{j,i} < d_{i,n+1}$, i.e., when a node accesses objects, local cache is preferred over other caches which are preferred over original data sources. The above definition of d is referred to as our access "price"[1] mode.

The cost of a node depends on where objects are placed and its access pattern. Due to cache space limitation, each node can cache only some objects locally and must decide which objects to place in its cache. Let S_i denote the cache size at node i ($S_i < m$) and P_i denote the set of objects cached at node i. Similarly, all other nodes decide which objects to place in their caches. The result is a global object placement $P = \{P_1, P_2, \ldots, P_n\}$. Then the cost of node i depends on the placement P. Let $C_i(P)$ denote the cost of node i under object placement P which is computed as follows:

$$C_i(P) = \sum_{k \in P_i} r_{ik} d_{i,i} + \sum_{\substack{k \notin P_i \\ k \in Q_{-i}}} r_{ik} d_{i,l(i,k)} + \sum_{\substack{k \notin P_i \\ k \notin Q_{-i}}} r_{ik} d_{i,n+1} \quad (1)$$

The cost of node i is the total access cost to serve requests for all the objects. The cost for node i to serve requests for any object k is the product of its request rate and the cost for node i to fetch object k which depends on the location from which the object is fetched, either from local cache, or from the "cheapest" nodes that caches object k or from the original source. Let $Q_{-i} = P_1 \cup \ldots \cup P_{i-1} \cup P_{i+1} \cup \ldots \cup P_n$ denote the set of objects collectively held by nodes other than node i under the global placement $P = \{P_1, P_2, \ldots, P_n\}$ and $d_{i,l(i,k)}$ denote the cost for node i to fetch the object from the "cheapest" node $l(i,k)$ that caches object k. More specifically, for each request for any object k, if object k is locally cached, it is accessed from local cache with cost $d_{i,i}$; otherwise if object k is cached at certain nodes in the NSCC group, it is accessed from the "cheapest" node $l(i,k)$ among those that store object k with cost $d_{i,l(i,k)}$; otherwise it is accessed from the original source with cost $d_{i,n+1}$.

Instead, under GL, objects at node i are sorted in descending order by their request rates and node i caches the S_i most popular objects. Then each object is accessed either from local cache or from its original source and the cost of node i under GL is computed as follows:

$$C_i(GL) = \sum_{k \le S_i} r_{ik} d_{i,i} + \sum_{k > S_i} r_{ik} d_{i,n+1} \quad (2)$$

NSCC seeks a guaranteed object placement P such that for each node in the group, its cost would be reduced as compared to that under GL. And the objective is formulated as follows:

$$\forall i,\ C_i(P) < C_i(GL) \quad (3)$$

which is the participation or individual rationality constraint for each rational node.

4 A Game-Theoretic Approach

As described in the former sections, NSCC nodes are intelligent rational decision-makers with both conflict and cooperation between them and these intelligent individuals interact with one another in an effort to achieve their own goals, which is a typical game. And thus we address this problem in a game-theoretic context, where nodes are the players. Each player implements a placement strategy that consists of choosing which objects to store locally in its limited storage space, at one or more occasions in the game. The goal of each player is to minimize its cost of serving all its requests for objects at the end of the game which is called *an NSCC game*.

Definition 1: (Best Response) Given a residual placement $P_{-i} = P - \{P_i\}$, the best response for node i is the placement $P_i \in A_i$ such that $C_i(P_{-i} + \{P_i\}) \le C_i(P_i + \{P_i'\})$, $\forall P_i' \in A_i$, where A_i is the set of available placements at node i.

The best response at node i is computed as follows: $g_{ik}(P_{-i})$ denotes the excess gain incurred by node i from replicating object k under P_{-i} and is defined as follows:

$$g_{ik}(P_{-i}) = \begin{cases} r_{ik}(d_{i,n+1} - d_{i,i}) & \text{for } k \notin Q_{-i} \\ r_{ik}(d_{i,l(i,k)} - d_{i,i}) & \text{for } k \in Q_{-i} \end{cases} \quad (4)$$

Objects are sorted in descending order by $g_{ik}(P_{-i})$ and the S_i most valuable objects are selected to cache, which is the best response at node i under P_{-i}.

Note that we assume that each node knows the exact object placements at other nodes when it computes its best response. The cases that either certain nodes are misinformed, or some nodes cheat about their placements are not considered since nodes are enforced to be frank or they will be kicked out from the NSCC group.

Definition 2: (Stable Placement) A global placement P is stable if and only if it is composed of individual placements that are best responses.

Therefore a stable placement P^* is just a *Nash equilibrium* [20] placement of an NSCC game in which no node can unilaterally change its placement to increase its gain such that additional stability achieves. And, above all, Nash equilibrium object placements satisfy the rational participation constraints that an NSCC game asks for. Subsection 4.1 discusses in what circumstances, there are Nash equilibrium object placements in NSCC games.

[1] Here, by price, we do not narrowly mean money.

Definition 3: (*Iterative Best Response* [IBR]) Given the initial global placement $P^{(0)}$ as that under GL, start an iterative procedure where at iteration (or call it round) l, any node i waits for its turn to play the game once and only once by performing the following steps:

Step 1: node i computes its best response $P_i^{(l)}$ to $P_{-i}^{(l,\ i-1)}$, after node i–1 and before node i+1;

Step 2: node i notifies others of its completeness and broadcasts its object placement changes (if there are) so that all nodes can compute $P^{(l,\ i)} = P_{-i}^{(l,\ i-1)} + \{P_i^{(l)}\}$.

$P^{(l,\ i-1)}$ is the global placement at iteration l after node (i–1)'s best response and prior to node i's best response; $P_{-i}^{(l,\ i-1)}$ is the corresponding residual placement with respect to node i. The IBR search stops and returns $P = P^{(t)}$ when at iteration t: $P^{(t)} = P^{(t-1)}$, i.e., when no node can profit by replacement.

We use IBR to identify the guaranteed object placement of an NSCC game. As mentioned, in each round of IBR, these selfish nodes are arranged in a certain order to play the game and each node should know its order so that no nodes would play the game simultaneously. We discuss that the ordering of playing the game impacts the resulting individual node costs, i.e., impacts the fairness among these selfish nodes, in performance evaluation section. The node ordering can be the same for all rounds of IBR and can be determined by some rules, e.g., based on the order of nodes' identifiers or their average access "prices" as described in the evaluation section. Or the node ordering can be variable for different IBR rounds and here we give a specific example about how these selfish nodes determine their ordering at each round: at each round of IBR including the procedure of exchanging the initial object placements under GL, each node generates a random *nonce* value and piggybacks the nonce value on to its object placement announcement. After receiving the n–1 nonce values from other nodes in this round, each node computes its turn in the next round based on the ranking of its nonce value among all these nonce values received from others (the range of nonce values should be large enough so that there is only a negligible chance that any two nodes are with equal nonce values).

Note that as a distributed algorithm, the IBR method has each node make its own object placement decisions based on its own local access pattern, and no exact access pattern needs to be exposed to others, which keeps both autonomy and some privacy for nodes.

4.1 The Resulting Object Placements

Regarding the resulting global object placement at each iteration of IBR, we can prove the followings:

Proposition 1. Each object in $P_1^0 \cup \ldots \cup P_n^0$ would be kept at least one copy in the NSCC group.

Proof. $\forall i, k_e \in P_i^0, k_i \notin P_i^0$ then $r_{ik_e} > r_{ik_i}$ such that object k_e is greedily cached at node i under GL. In the following IBR rounds, if object k_e is replaced by k_i at certain step, then $r_{ik_e}(d_{i,l(i,k_e)} - d_{i,i}) < r_{ik_i}(d_{i,l(i,k_i)} - d_{i,i})$. Since $r_{ik_e} > r_{ik_i}$, then $d_{i,l(i,k_e)} < d_{i,l(i,k_i)}$. If $d_{i,l(i,k_e)} = d_{i,n+1}$, i.e., k_e at node i is the single copy in the NSCC group before the replacement, then $d_{i,l(i,k_i)} > d_{i,n+1}$ which contradicts our assumption that $\forall i, j, d_{i,j} < d_{i,n+1}$. That said, an object in $P_1^0 \cup \ldots \cup P_n^0$ can be evicted only if there are replicas of it in the NSCC group. And thus each object in $P_1^0 \cup \ldots \cup P_n^0$ would have at least one copy in the NSCC group.

Proposition 2. At most $\min\{\sum_{i=1}^{n} S_i - |P_1^0 \cup \ldots \cup P_n^0|, m - |P_1^0 \cup \ldots \cup P_n^0|\}$ objects can be inserted into the NSCC group during IBR iterations.

Proof. From proposition 1, we know that objects in $P_1^0 \cup \ldots \cup P_n^0$ would take at least $|P_1^0 \cup \ldots \cup P_n^0|$ storage space and at most $\sum_{i=1}^{n} S_i - |P_1^0 \cup \ldots \cup P_n^0|$ space can be left for additional objects to insert into. And therefore at most $\min\{\sum_{i=1}^{n} S_i - |P_1^0 \cup \ldots \cup P_n^0|, m - |P_1^0 \cup \ldots \cup P_n^0|\}$ objects can be inserted into the NSCC group during these IBR iterations.

Regarding that whether an NSCC game would converge to a Nash equilibrium placement, we have the following statement:

Proposition 3. A Nash equilibrium object placement is guaranteed to find in polynomial time if the access price model forms an ultra-metric. Otherwise, it is NP-complete to determine the existence of a Nash equilibrium object placement.

Proof. Gopalakrishnan et al. proved proposition 3 in [17].

That the access "price" function forms an ultra-metric means that $\forall i, j, k, d_{i,k} \leq \max\{d_{i,j}, d_{j,k}\}$. More specifically, in the ultra-metric space, it is preferable for a node to access another node in the NSCC group via their direct link rather than that via a third node. Of course, we prefer Nash equilibrium object placements as they satisfy the rational participation constraints of all nodes, and meanwhile, offer additional stability. Actually in all our simulations, we find Nash equilibrium object placements (after at most nine rounds). But it is cost reduction rather than the stability that promotes the cooperation among these selfish nodes. Even if Nash equilibrium object placements do not exist, we can enforce that nodes cannot deviate from the committed object placement. Otherwise they will be kicked out from the NSCC group so that they have no incentive to deviate as their losses outweigh gains in the long term. In the next subsection, an example shows that with IBR method, sometimes certain node(s) may not be rational, but additional condition is placed on the IBR such that nodes in the NSCC group would be always rational.

4.2 Individual Rationality

In this subsection, we examine in which cases the rational participation constraint can be satisfied for all selfish nodes in the group. We first demonstrate that, with an example and comparing TSLS with NSCC, in a scenario where the access "prices" of different node pairs differ, the participation constraints of certain nodes may be violated in the object placement obtained from TSLS [5] which ignores the access "price" difference.

Example 1. Figure 2 shows an NSCC group comprised of three selfish nodes indexed by 1, 2, 3, each with caching capacity of $S_1 = 1$, $S_2 = 2$, $S_3 = 1$. The set of objects that may be requested by these nodes is {1,2,3,4} and the request rates[2] at the three nodes are $r_1 = r_2 = r_3 =$ {0.5,0.25,0.15,0.10}. We assume for $i = 1,2,3$, $d_{i,i} = 0$, $d_{i,4} = 100$, and $d_{1,3} = 1$, $d_{1,2} = d_{2,3} = x$ is a variable within the range of [1,60].

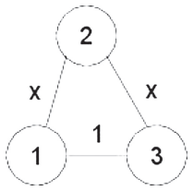


Figure 2 The Topology of Example 1

Under GL, the object placements at the three nodes are $P_1 = P_3 = \{1\}$, $P_2 = \{1,2\}$. Under TSLS ($t_l = 0$, $t_r = 1$, $t_s = 100$), if these nodes play according to their indices, the resulting placement is $P_1 = \{3\}$, $P_2 = \{2,4\}$, $P_3 = \{1\}$. However, if x, the costs for node 2 to fetch data from node 1 or from node 3 are large, this may cause the violation of the participation constraint of node 2. This is indeed as shown in Figure 3, where for $38.46 < x < 60$, the cost of node 2 under TSLS is greater than that under GL. In contrast, under NSCC, while $P_1 = \{3\}$, $P_3 = \{1\}$ hold all the time, for $1 \le x < 20$, $P_2 = \{2,4\}$ where the cost of node 2 coincides with that under TSLS; but for $20 < x < 60$, $P_2 = \{1,2\}$, where its cost is still smaller than that under GL.

The previous example shows that in the scenario where the costs of accessing an object between different nodes vary and the rationality of certain node is not satisfied under TSLS, NSCC offers alternative object placement such that the rationality is guaranteed.

But one also can easily construct examples in which, at some iteration of IBR, the rational participation constraints of certain nodes in NSCC are not satisfied, i.e., nodes are not always rational.

Example 2. Given three selfish nodes indexed by 1, 2, 3, with cache capacity $S_1 = S_2 = S_3 = 3$, and five distinct objects {1,2,3,4,5} that may be requested by them. The topology of the three nodes are shown in Figure 4 and for $i = 1, 2, 3$, $d_{i,i} = 1$, $d_{i,4} = 12$ and $d_{1,2} = 5$, $d_{1,3} = 10$. The access patterns at the three nodes are $r_1 =$ {0.32,0.28,0.22,0.11,0.07}, $r_2 =$ {0.34,0.20,0.17,0.16,0.13}, and $r_3 =$ {0.33,0.27,0.20,0.13,0.07}.

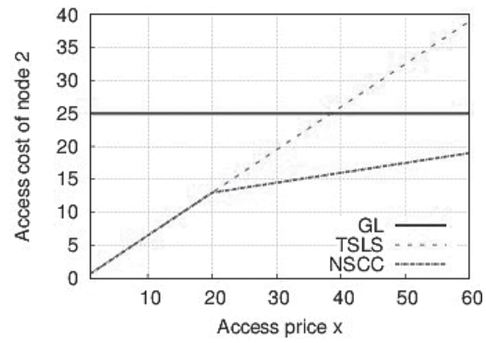


Figure 3 The Cost of Node 2 under Different Object Placement Strategies in Example 1

Figure 4 The Topology of Example 2

If under GL, each node chooses its most popular objects to store and thus $P_1 = P_2 = P_3 = \{1,2,3\}$. And the resulting costs at the three nodes are 2.98, 4.19 and 3.20 respectively.

When following the IBR method and nodes play the game according to their indices, then after one round, the object placement would be $P_1 = \{1,2,4\}$, $P_2 = \{1,2,5\}$, $P_3 = \{1,3,4\}$ and the resulting individual node costs are 3.26, 2.32 and 2.36 separately. Since 3.26 > 2.98, node 1 is not rational at all. Therefore, IBR cannot guarantee nodes are always rational.

Nevertheless, the following definitions give additional condition on IBR such that nodes are always rational.

Definition 4: (Possible Maximum Eviction Loss) For an object $k \in P_i^0$, at iteration 1, node i computes the possible maximum eviction loss of object k based on $P_{-i}^{(l,\, i-1)}$ as follows:

$$pmel_{ik}^{1}(P_{-i}^{(1,i-1)}) = \begin{cases} r_{ik}(d_{ik}^{\min} - d_{i,i}) & \text{if } k \in \cup_{j<i} P_j^1 \\ r_{ik}(d_{ik}^{\max} - d_{i,i}) & \text{elseif } k \in \cup_{j>i} P_j^0 \\ r_{ik}(d_{i,n+1} - d_{i,i}) & \text{otherwise} \end{cases} \quad (5)$$

where $d_{ik}^{\min} = \min\{d_{i,j} \mid j < i,\ k \in P_j^1\}$ denoting the minimal cost of fetching it from nodes (before node i) that have committed to cache object k, and $d_{ik}^{\max} = \max\{d_{i,j} \mid j > i,\ k \in P_j^0\}$ representing the maximal cost of fetching it from nodes after node i. The possible maximum eviction loss

[2] Note that the request rates in this example are not normalized.

should be the product of its request rate and the difference between the cost of fetching it from the possible most "expensive" node and that from local cache. And the possible most "expensive" node is chosen as follows. If some of the first i-1 nodes have already committed to host object k (i.e., $k \in \cup_{j<i} P_j^1$), the possible most "expensive" node would be the "cheapest" node that commits to hold it. Otherwise if object k is cached at some nodes after node i (i.e., $k \notin \cup_{j<i} P_j^1$, $k \in \cup_{j>i} P_j^0$), the possible most "expensive" node would be the most "expensive" node (after node i) that caches object k as these nodes after node i may evict object k, but it can guarantee that at least one copy would be kept in the NSCC group as proved in proposition 1. Otherwise, the possible most "expensive" node is indeed the original source.

Definition 5: (Possible Minimum Insertion Gain) For an object $k \notin P_i^0$, at iteration 1, node i computes the possible maximum insert gain of object k based on $P_{-i}^{(l,\, i-1)}$ as follows:

$$pmig_{ik}^1(P_{-i}^{(1,i-1)}) = r_{ik}(\min\{d_{ik}^{(i-1,\min)}, d_{ik}^{(i+1,\min)}\} - d_{i,i}) \quad (6)$$

where $d_{ik}^{(i-1,\min)} = \min\{d_{i,j} \mid j < i,\, k \in P_j^1\}$ representing the minimal cost of fetching it from those that have committed to cache it, and $d_{ik}^{(i+1,\min)} = \min\{d_{i,j} \mid j > i\}$ denoting the minimal cost of fetching it from those (after node i) that have the potential to cache it. The possible minimum insertion gain should be the product of its request rate and the difference between the cost of fetching it from the possible "cheapest" node and that from local cache. The possible "cheapest" node is chosen among those that have committed to cache it and those that have the potential to cache it. And for nodes after node i, any may choose to cache object k.

Definition 6: (Conservative Response) For the one round game starting from the object placement under GL, a node would replace an object in its cache if and only if the minimum possible insertion gain of the inserted object is larger than the maximum possible eviction loss of the replaced one.

Proposition 4. If each node gives conservative response and the NSCC game plays one round, each node is rational in the resulting object placement.

Proof. The proposition is proved by the definition of conservative response since the conservative responses guarantee that at each node, each replacement results in cost reduction anyway.

4.3 Potential Gain of a Node by Playing Again

The potential gain of a node by playing again is defined as the cost reduction of the node if it plays again after a few rounds of IBR and it serves as the incentive for the node to play again.

A node may choose to play again if some of its local cached objects are cached in certain subsequent nodes to which the access "prices" from this node are "cheap" and thus the node would benefit by evicting such objects; or if some objects just evicted by this node are also evicted in certain subsequent nodes to which the access "prices" from this node are "cheap" and thus the node would benefit by reinserting such objects.

Meanwhile, the aforementioned two cases also serve as the reasons that some node(s) may lose after playing again as even though the node(s) makes the right placement decision(s) based on the current residual object placement, the actions (eviction and insertion) of subsequent nodes may cause the decisions made by these nodes playing first to be unreasonable.

In the performance evaluation section, with numerical examples, we show that only a few nodes benefit from playing again after the first round and the gains are small as compared to their costs. That said, nodes have little incentive to play again after the first round.

4.4 Complexity Analysis

In NSCC, each participant takes part in the decision-making process, which convinces these selfish nodes. We analyze the corresponding time and space complexity of the NSCC game in this section.

Firstly, each node has to determine its initial object placement by sorting the request rates of objects which takes time $O(mlgm)$. This specific decision making at a node does not rely on anything at other nodes and thus these nodes can make the decisions in parallel.

Then in each round of IBR, any node i should compute the insertion gain for each object and sort these gains in descending order, which takes time $O(m+mlgm)$ altogether. And then if the local object placement is changed, the node should announce the changes to other nodes. If in NDN where multicast is naturally supported, it may be enough that the node just sends the determined object placement once which takes time $O(S_i)$; otherwise if in IP networking, the node may have to send its determined object placement to any other participants which takes time $O((n-1)S_i)$. So in each round, the time that node i takes is bounded by $O(m+mlgm)+(n-1)S_i) = O(nm+mlgm)$. As nodes play the game one by one, these nodes take time $O(n(nm+mlgm)) = O(n^2m+nmlgm)$ altogether. And thus the time complexity would be $O((mlgm+N(n^2m+nmlgm))$ where N is the number of IBR rounds that the NSCC game plays. In all our numerical examples, the algorithm stops after at most nine rounds, and as discussed in Subsection 4.3, these selfish nodes have little incentive to play again after the

first round, so the NSCC game may practically take time $O(mlgm+n^2m+nmlgm)$.

The space consumption at any node i is comprised of three components: the storages of the access "prices" $d_{i,j}$s from this node to other nodes in the group, the global object placement P, and the insertion gain values of objects $g_{ik}(P_{-i})$s, which take space $n-1$, $\sum_{i=1}^{n} S_i$ and m separately. And therefore the required memory space at each node is bounded by $O(n-1+nm+m)$.

5 Performance Evaluation

In this section, we study how NSCC improves access costs of selfish nodes in an NSCC group with numerical examples. We are interested in analyzing cases where these selfish nodes have similar access patterns (object preferences) so that they can mutually benefit by implicit cooperation. Many measurement studies have observed heavy-tailed or Zipf distributions (i.e., the i^{th} popular object has a request probability proportional to $1/i^s$ where s is the Zipf preference) in request popularities (e.g., [21-22]). Our popularity statistics of web request traffic collected from the edge of *JiangSu Education and Research NETwork* (JSERNET) on March 27, 2012 as illustrated in Figure 5 and Fayazbakhsh et al.'s statistics of request logs collected from three CDN vantage points [23] reconfirm such heavy-tailed behavior in recent workloads. So we assume the access pattern at any node i follows Zipf distribution with exponent s in our simulations. Without loss of generality, we assume all caches have the same cache capacity in terms of the number of unit-sized objects for all our simulations.

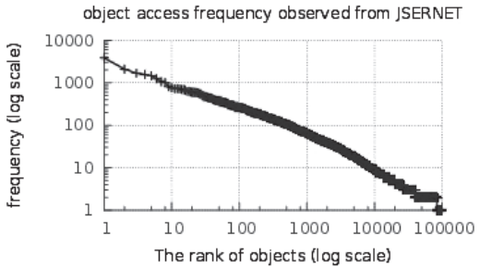


Figure 5 Request Popularity Distribution of Web Requests from JSERNET

To be realistic, in Subsections 5.1 and 5.2, we conduct simulations on the PoP-level topology of AS 209 (58 PoP nodes and 108 links) from [24] (we treat PoP nodes in the underlying network as nodes with caching capability and they are competitive and selfish) and delays on links are treated as the access "prices" between PoP nodes. Figure 6 shows the Cumulative Distribution Function (CDF) of access "prices" between PoP nodes and it can be seen that 90% node pairs have access "prices" smaller than 50ms and 99.8% node pairs have access "prices" smaller than 90ms. And we set the access "prices" from any node i = 1, 2, …, 58 to original content sources, i.e., $d_{i,59}$, to be 130 ms for simulations conducted in this topology.

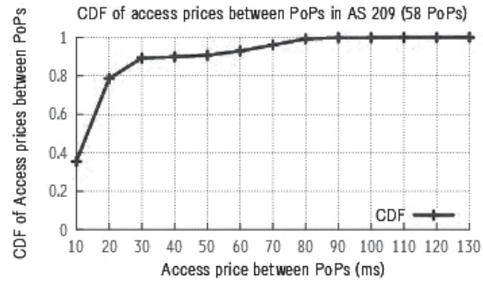


Figure 6 The CDF of Access "Prices" between PoPs in AS 209

We compare the costs of nodes under NSCC against that under GL and against that under TSLS (under TSLS, t_s is set to the average of the access "prices" for any caches to access data from other caches). To study the gain of playing again in NSCC, we show the costs after one round of IBR and that after multirounds.

5.1 Impact of Access Patterns at Nodes

For the evaluation of the impact of access patterns, to draw more insightful conclusions, this work considers the following three cases of nodes' access patterns [25]:

- Case 1: the access patterns at different nodes follow the same Zipf distribution with exponent s and the rank of objects remains the same for all nodes.
- Case 2: the rank of requested objects at different nodes remains the same, but the access patterns follow Zipf distribution with different exponents: they become more concentrated around the most popular objects as the node index i increases.
- Case 3: the access patterns at different nodes follow the same Zipf distribution but the ranking of a given object changes for different nodes (i.e., the objects at a specific ranking at nodes differ).

In the three cases, we set that there are 1,000 unit-sized objects that may be requested by these nodes and each node is with a caching capacity of 10 objects. And the result for each specific simulation is averaged over 500 runs, where at each run a random permutation of the order of play is selected.

5.1.1 Case 1

We conduct simulations with different Zipf exponents s from 0.6 to 1.5 [21][26] and Figure 7(a) shows the average access cost over all nodes under different placement strategies. It can be seen that, in different scenarios, TSLS and NSCC both reduce the average node access costs as

compared to that under GL (on average, by 39.91% and 67.67% separately) and NSCC further reduces the average node access costs by on average, 27.76% from that under TSLS. Hence, compared to TSLS, NSCC is more suitable for enabling selfish nodes to cooperate in caching when the access "prices" of different nodes pairs are various and they have the same access patterns. And the reason should be that under TSLS, when a node makes placement decisions, it does not realize that some popular objects are stored at nodes from which too "expensive" to fetch data such that they should be locally cached.

Moreover, as the Zipf exponent s increases, even though the caching capacity of the group does not change, the average node access costs under different placement strategies are all reduced and the rates of average node access cost reduction decrease. This is because the access patterns become heavier-tailed with the increase of the Zipf exponent s such that the frequencies at which these nodes access the objects that are not cached in the group reduce, and the reduction of the frequencies becomes smaller and smaller.

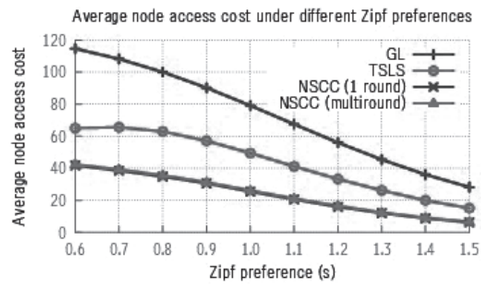
5.1.2 Case 2

The request rates over the objects k = 1, 2, …, m at different nodes are drawn from Zipf distributions with different exponents s (within the range of [0.6, 1.0]). More particularly, we set that node 1 has access pattern following Zipf distribution with exponent s = 0.6. Then, for node i = 2, 3, …, n, s is increased by $p(i–1)$ and $p = (1.0 – 0.6) / (n–1)$. The results of average access cost of each node (over 500 runs) are illustrated in Figure 7(b). As shown, the node access costs under TSLS and under NSCC are reduced as compared to that under GL (on average, by 41.72% and 64.35%) and NSCC outperforms TSLS. And as node index increases, i.e., as Zipf preference s increases, the node access costs primarily present a trend of decreasing since nodes' access patterns become heavier and heavier-tailed. However, at nodes 23, 26, 30 and 58, in the lines of TSLS and NSCC, spikes are seen. We dig into the topology data and find out that the average access "prices" to other nodes from the four nodes 58, 23, 26, 30 are the four most "expensive" which are higher than others by multiple times (up to 2.5 to 6 times). That said, the high access "prices" from the four nodes to others contribute to their large costs.

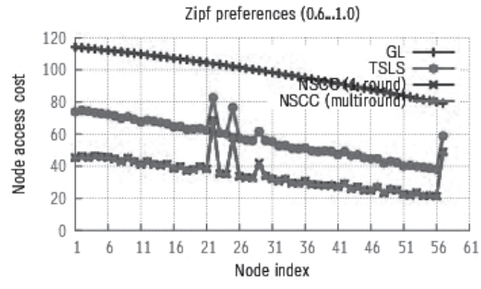
5.1.3 Case 3

The access patterns at all nodes follow Zipf distribution with a typical exponent s = 0.73 (see e.g., [21][26]). In order to establish dissimilarity in the nodes' interests, the rank of objects at node 1 is assigned [1, 2, …, m] (m = 1,000) and this rank is shifted to the left by different positions for each of the other nodes. In general, the rank of objects at node i, i = 2, …, n (n = 58) is shifted by $k(i–1)$ positions, where k is the shift parameter. For example, when k = 1, the ranking of objects at node 2 is [2, 3, …, m, 1], that at node 3 is [3, 4, …, m, 1, 2], and so on. We consider different values of the shift parameter k ($k(n–1) < m$, then k = 1, 2, …, 17) and the average node access costs under different shift parameters are shown in Figure 7(c).

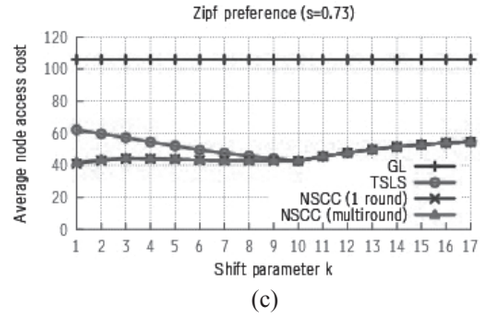
It can be seen that the average node access costs under TSLS and under NSCC are reduced as compared to that under GL (on average, by 51.59% and 56.31% separately). Under NSCC, as the shift parameter k increases, when $k \leq 10$, the average node access cost hardly changes, but when $k > 10$, it tends to increase. The reason is that as the dissimilarity in the nodes' interest increases to a certain degree, nodes tend to cache their own most popular objects and less and less popular objects can be shared between nodes such that nodes have to access less popular objects from original content sources. This demonstrates that NSCC performs better when nodes' access patterns are more similar. Another interesting observation is that as the



(a)



(b)



(c)

Figure 7 Average Node Access Costs under Different Placement Strategies and with Different Access Patterns

dissimilarity in the nodes' interest increases, the average node access cost under TSLS gradually approaches that under NSCC and later they overlap. This is because as the dissimilarity in the nodes' interests increases, less and less popular objects can be shared between them and the access patterns play the decisive role in the object placement decision processes at these nodes under NSCC obscuring the impact of TSLS's unawareness of large distances between some nodes.

5.2 Impact of Cache Capacity

In this subsection, we continue to conduct simulations on the PoP level topology of AS 209. Likewise, we assume there are 1,000 unit-sized objects that may be requested, the access patterns at different nodes follow the same Zipf distribution with exponent $s = 0.73$ and the rank of requested objects remains the same for all the nodes. In different simulations, the cache sizes vary from a capacity of 20 objects to that of 100 objects. The average node access costs in different simulations are illustrated in Figure 8 and the result for each simulation is averaged over 500 runs, where at each run a random permutation of the order of play is selected.

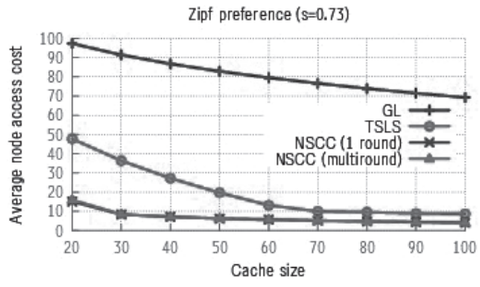


Figure 8 Average Node Access Costs under Different Placement Strategies and with Different Cache Sizes

As shown, in each simulation, NSCC performs best among the three placement strategies and reduces the average node access cost (on average) by 88.18% as compared to under GL, a significant gain that attracts nodes to join NSCC group (TSLS accordingly reduces the average node access cost on average by 70.46% as compared to under GL). And as the cache sizes of nodes increase, the average node access costs under different placement strategies are all reduced and the rates of reduction decrease since while more objects can be cached in the group, these additionally cached objects are accessed by nodes with smaller and smaller frequencies. Moreover, under NSCC, when cache sizes of nodes are larger than 30 (the caching capacity of the NSCC group is $30 \times 58 = 1{,}740 > 1{,}000$ objects), the average node access costs are not reduced any more as all requested objects are already cached in the group and more storage space would not make much difference.

5.3 Impact of the Order Of Play

In this subsection, we study the impact of the order of play on individual node access costs. The studied NSCC group is comprised of five selfish nodes and their topology and access "prices" between each other (we choose "cheapest" path between two nodes) are shown in Figure 9 (note that $\forall i = 1, 2, \ldots, 5$, $d_{i,6} = 100$). We set there are 50 objects in the system and the caching capacity of each node is 10 objects. And we assume the access patterns at all the nodes follow the same Zipf distribution with exponent $s = 0.73$ and the rank of these objects remains the same at all five nodes.

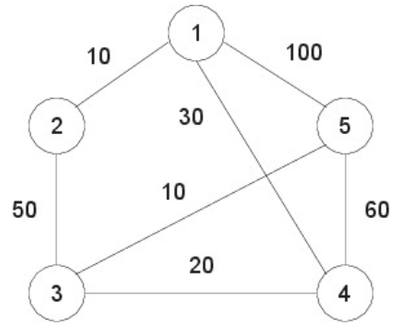
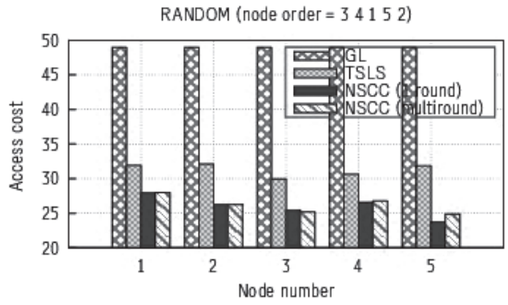


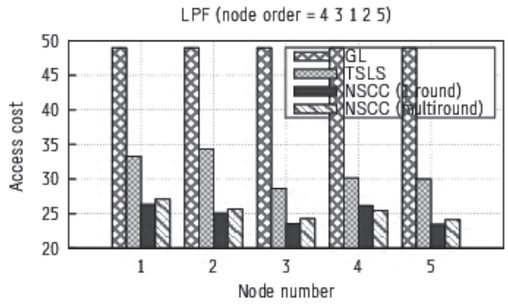
Figure 9 The Topology of an NSCC Group Comprised of Five Selfish Nodes

The access costs of individual nodes under different placement strategies and with three different node orderings are evaluated (under TSLS, t_s is set to the average access "prices" of all node pairs) and are illustrated in Figure 10. For each node i, we compute its average access "price" from other nodes like this $avg\_D_i = \frac{1}{N-1} d(i, j)$. The studied three node orderings are (a) random order, (b) the node with Least average access "Price" play First (LPF), and (c) the node with Most average access "Price" play First (MPF) (note that the ordering is the same at each round of NSCC).

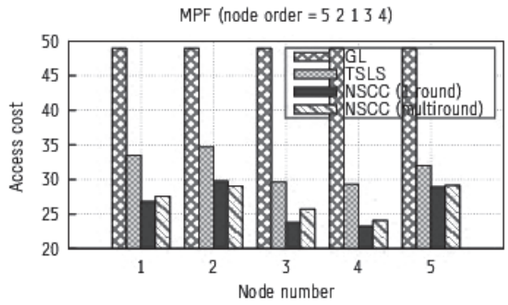
In the three orderings, both TSLS and NSCC outperform GL policy, producing smaller access costs at individual nodes (on average, the individual node access costs are reduced by 35.77% and 47.32% separately) such that participation constraints are satisfied; and NSCC still outperforms TSLS. It can be computed that after multirounds, the average access costs of these nodes with random order, LPF and MPF are 26.21, 25.31 and 27.12 cost units respectively; the maximum access costs among these five nodes with random order, LPF and MPF are 27.98, 27.07 and 29.15 separately; and the variance of individual node access costs with random order, LPF and



(a) Random Order



(b) LPF Order



(c) MPF Order

Figure 10 Individual Node Access Costs under Different Placement Strategies and with Different Node Orderings

MPF are 6.31, 5.72 and 19.23 respectively. The above statistics suggest that among the three orderings, with the LPF ordering, the NSCC game performs the best since nodes tend to serve its requests with least access costs and the node with the maximum cost (i.e., with least incentive to cooperate) is the happiest among the three orderings, i.e., nodes are treated in the fairest way with LPF ordering in this simulation. The reason is that under the LPF ordering, the NSCC game results in that objects with high request rates are cached at nodes which pay higher average access "prices" such that the access cost of such nodes would be reduced; meanwhile, the subsequent popular objects are cached at nodes starting from the one with the least average access "price" to the one with the largest average access "price" such that these subsequent popular objects can be accessed by nodes with "cheaper price." In contrast, with the MPF ordering, the most popular objects are stored at the node with the least average access "price" and the subsequent popular objects are cached at nodes starting from the one with the largest average access "price" to the one with the least average access "price."

Another interesting observation shown in Figures 7, 8 and 10 is that the gains of playing again are negligible as compared to the access costs after the first round; and in Figure 10, by playing again, the individual access costs of some nodes are reduced, but at the expense of others. The potential gain of each node by playing again with different orderings under NSCC strategy in Figure 10 is shown in Table 1. With the three orderings, only nodes playing first benefit by playing again (Nodes 4, 2 and 3 respectively) at the expense of others. It can be seen that the gains by playing again are small as compared to the cost values and thus nodes have little incentive to play again (the gains by playing again may not even cancel out the cooperation overhead in playing again).

Table 1 Potential Gain of Each Node by Playing Again after the 1st Round

Node	1	2	3	4	5
LPF	0	0	0	0.67	0
MPF	0	0.73	0	0	0
random order	0	0	0.25	0	0

6 Conclusion & Future Work

This work proposes a scheme that enables selfish nodes to cooperate in caching, here dubbed *Not So Cooperative Caching* (NSCC). We consider a network comprised of selfish nodes; each is with caching capability and an objective of reducing its own access cost by fetching data from local cache or from other caches. The challenge is to determine what objects to cache at each node so as to induce low individual node access costs, and the realistic access "price" model which allows various access "prices" of different node pairs further complicates the decision making process.

Using a game-theoretic approach -- Iterative Best Response (IBR), and taking the various access "prices" into account, NSCC seeks a global object placement in which the access costs of individual selfish nodes would be reduced as compared to that if they operate in isolation so as to incur implicit cooperation even among these selfish nodes. In the distributed algorithm IBR, each node makes its own placement decisions based on its local access pattern, the object placements at other nodes and the access "prices" from this node to other nodes. We analyze the

performance of NSCC both in theory and with extensive numerical examples. Although individual node rationality (i.e., not loosing by participation) is not guaranteed, NSCC achieves such rationality easier relative to other placement strategies. Furthermore, if each node gives conservative response in IBR method, individual node rationality can be improved. Our extensive experiments show that in almost every case, NSCC reduces individual node access costs as compared to the greedy local or previous work (TSLS) that ignores the difference in access "prices." NSCC performs better when nodes follow more similar access patterns and is more efficient when users' access patterns are less heavy-tailed. Finally, NSCC allows for a fairer treatment of nodes according to their average access "prices" to other nodes.

Our next step is to further extend our work to the cases that allow nodes in the group fail with some probability, which is common in the network environment and thus is further anchored in reality, and to show how the scheme can be implemented in Information Centric Networking such as Named Data Networking featuring routing by name, multipath routing and in-network caching despite node failure and node cheating.

Acknowledgements

We are grateful to anonymous reviewers of Journal of Internet Technology. This work was conducted under the support of Jiangsu Key Laboratory of Computer Networking Technology and the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education. And this work was sponsored by the National Grand Fundamental Research 973 program of China under Grant No. 2009CB320505, the National Nature Science Foundation of China under Grant No. 60973123, and the Technology Support Program (Industry) of Jiangsu under Grant No. BE2011173. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of those sponsors.

References

[1] Jussi Kangasharju, James Roberts and Keith W. Ross, *Object Replication Strategies in Content Distribution Networks*, *Computer Communication*, Vol.25, No.4, 2001, pp.376-383.

[2] Madhukar R. Korupolu and Michael Dahlin, *Coordinated Placement and Replacement for Large-Scale Distributed Caches*, *IEEE Transactions on Knowledge and Data Engineering*, Vol.14, No.6, 2002, pp.1317-1329.

[3] Sem Borst, Varun Gupta and Anwar Walid, *Distributed Caching Algorithms for Content Distribution Networks*, *Proc. INFOCOM'10*, San Diego, CA, March, 2010, pp.1478-1486.

[4] Jason-Min Wang, Jun Zhang and Brahim Bensaou, *Intra-AS Cooperative Caching for Content-Centric Networks*, *Proc. of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, Hong Kong, China, August, 2013, pp.61-66.

[5] Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos and Ioannis Stavrakakis, *Distributed Selfish Replication*, *IEEE Transactions on Parallel and Distributed Systems*, Vol.17, No.12, 2006, pp.1401-1413.

[6] Mark Gritter and David R. Cheriton, *An Architecture for Content Routing Support in the Internet*, *Proc. of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems*, San Francisco, CA, March, 2001, pp.37-48.

[7] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye-Hyun Kim, Scott Shenker and Ion Stoica, *A Data-Oriented (and beyond) Network Architecture*, *ACM SIGCOMM Computer Communication Review*, Vol.37, No.4, 2007, pp.181-192.

[8] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs and Rebecca L. Braynard, *Networking Named Content*, *Proc. of CoNEXT'09*, Rome, Italy, December, 2009, pp.1-12.

[9] Jarno Rajahalme, Mikko Särelä, Pekka Nikander and Sasu Tarkoma, *Incentive-Compatible Caching and Peering in Data-Oriented Networks*, *Proc. of CoNEXT'08*, Madrid, Spain, December, 2008, pp.1-6.

[10] Enric Herrero, José González and Ramon Canal, *Distributed Cooperative Caching: An Energy Efficient Memory Scheme for Chip Multiprocessors*, *IEEE Transactions on Parallel and Distributed Systems*, Vol.23, No.5, 2012, pp.853-861.

[11] Ji-Chuan Chang and Gurindar S. Sohi, *Cooperative Caching for Chip Multiprocessors*, *Proc. of ISCA'06*, Boston, MA, June, 2006, pp.264-276.

[12] Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou and John Kubiatowicz, *Selfish Caching in Distributed Systems: A Game-Theoretic Analysis*, *Proc. of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, Newfoundland, Canada, July, 2004, pp.21-30.

[13] Eva Jaho, Ioannis Koukoutsidis, Ioannis Stavrakakis and Ina Jaho, *Cooperative Content Replication in Networks with Autonomous Nodes*, *Computer Communication*, Vol.35, No.5, 2012, pp.637-647.

[14] Eva Jaho, Ioannis Koukoutsidis, Ioannis Stavrakakis and Ina Jaho, *Cooperative Replication in Content Networks with Nodes under Churn*, *Proc. of NETWORKING'08*, Singapore, May, 2008, pp.457-469.

[15] Avraham Leff, Joel L. Wolf and Philip S. Yu, *Replication Algorithms in a Remote Caching Architecture*, *IEEE Transactions on Parallel and Distributed Systems*, Vol.4, No.11, 1993, pp.1185-1204.

[16] Gerasimos G. Pollatos, Orestis A. Telelis and Vassilis Zissimopoulos, *On the Social Cost of Distributed Selfish Content Replication*, *Proc. of NETWORKING'08*, Singapore, May, 2008, pp.195-206.

[17] Ragavendran Gopalakrishnan, Dimitrios Kanoulas, Naga Naresh Karuturi, C. Pandu Rangan, Rajmohan Rajaraman and Ravi Sundaram, *Cache Me If You Can: Capacitated Selfish Replication Games*, *Proc. of 10th Latin American Symposium*, Arequipa, Peru, April, 2012, pp.420-432.

[18] Ozgur Ercetin and Leandros Tassiulas, *Market-Based Resource Allocation for Content Delivery in the Internet*, *IEEE Transactions on Computers*, Vol.52, No.12, 2003, pp.1573-1585.

[19] Panayotis Antoniadis, Costas Courcoubetis and Robin Mason, *Comparing Economic Incentives in Peer-to-Peer Networks*, *Computer Networks*, Vol.46, No.1, 2004, pp.133-146.

[20] Martin J. Osborne and Ariel Rubinstein, *A Course in Game Theory*, MIT Press, Boston, MA, 1994.

[21] Lee Breslau, Pei Cao, Li Fan, Graham Phillips and Scott Shenker, *Web Caching and Zipf-Like Distributions: Evidence and Implications*, *Proc. of INFOCOM'99*, New York, March, 1999, pp.126-134.

[22] Phillipa Gill, Martin Arlitt, Zongpeng Li and Anirban Mahanti, *Youtube Traffic Characterization: A View from the Edge*, *Proc. of IMC'07*, San Diego, CA, October, 2007, pp.15-28.

[23] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce M. Maggs, K. C. Ng, Vyas Sekar and Scott Shenker, *Less Pain, Most of the Gain: Incrementally Deployable ICN*, *Proc. of SIGCOMM'13*, Hong Kong, China, August, 2013, pp.147-158.

[24] Neil Spring, Ratul Mahajan and David Wetherall, *Measuring ISP Topologies with Rocketfuel*, *Proc. of SIGCOMM'02*, Pittsburgh, PA, August, 2002, pp.133-145.

[25] Eva Jaho, Merkouris Karaliopoulos and Ioannis Stavrakakis, *Social Similarity as a Driver for Selfish, Cooperative and Altruistic Behavior*, *Proc. of the 2010 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Montreal, Canada, June, 2010, pp.1-6.

[26] Hazem Gomaa, Geoffrey Messier, Robert Davies and Carey Williamson, *Media Caching Support for Mobile Transit Clients*, *Proc. of the 2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Marrakech, Morocco, October, 2009, pp.79-84.

Biographies

Xiaoyan Hu, a PhD candidate in School of Computer Science and Engineering, Southeast University, focuses her research interests on information centric networking, in-network caching and scalable name-based routing. She received her BS in software engineering from Nanjing University of Science and Technology, and MS in computer architecture from Southeast University.

Jian Gong is a professor in School of Computer Science and Engineering, Southeast University. His research interests are network architecture, network intrusion detection, and network management. He received his BS in computer software from Nanjing University, and his PhD in computer science and technology from Southeast University.