Data Caching in Ad Hoc Networks Using Game-Theoretic Analysis

Yutian Chen

Economics Department
California State University at Long Beach
Long Beach, CA 90840
ychen7@csulb.edu

Hoang Dang and Bin Tang

Electrical Engineering and Computer Science Department
Wichita State University
Wichita, KS 67260
hoangdangninh@gmail.com, bintang@cs.wichita.edu

Abstract— **Extensive research has been performed to study selfish data caching in ad hoc networks using game-theoretic analysis. However, due to the caching problem's theoretical root in classic facility location problem and k-median problem, most of the research assumes i), the data items are initially outside of the network, and ii), the caching cost is either a constant or not considered. In this paper, we study a general data caching model in which the data item is initially in the network, and both caching and access cost are distance-dependent in multi-hop ad hoc networks. We first show the studied problem is NP-hard. We construct a pure Nash Equilibrium, in which a node will not deviate its caching strategy if others remain theirs. However, a Nash Equilibrium may not guarantee social optimal cost – due to the selfishness of each node, the price of anarchy, which is the relative cost of the lack of cooperation among nodes, could be as large as $O(N)$, where N is number of nodes in the network. Using an external incentive mechanism based upon a payment model, we construct a Nash Equilibrium wherein social optimal is also achieved.**

Keywords – Data caching; game theory; ad hoc networks

I. Introduction

Ad hoc networks are multi-hop wireless networks consisting of small wireless computing devices such as conventional computers (e.g., PDA, laptop, or PC), or embedded processors such as tiny, low-cost, and low-power sensor motes. Ad hoc networks are constructed mainly for the information sharing and task coordination among a group of people, without the support of any communication infrastructure. For example, in an ad hoc network established for spontaneous meeting, several authors can meet and coordinate to modify the same document (e.g., an article or a powerpoint slides) in a distributed fashion. Similarly, in interconnected distributed information systems, an object (a web page, an image, a video clip, or a file) may be accessed from multiple distributed locations (network nodes) simultaneously.

Caching has been proposed to be an effective technique to facilitate information access in ad hoc networks. Besides the traditional advantages brought by caching such as less data access latency, improved data reliability and fault tolerance, utilizing caching to optimize network performance of ad hoc networks is motivated by the following two aspects. First, the ad hoc networks are multi-hop networks. Thus, remote access of information typically occurs via multi-hop routing, wherein access latency can be particularly reduced by data caching. Second, ad hoc networks are generally resource constrained in terms of wireless bandwidth, memory capacity and battery energy of nodes. Data caching can help reduce communication cost among nodes, which results in conserving battery energy and minimizing bandwidth usage in ad hoc networks.

Due to above reasons, many caching techniques have been developed recently to achieve good overall performance of the ad hoc network [4, 5, 18, 20, 21] (please refer to Section II for a comprehensive literature review). They are all cooperative caching techniques, wherein nodes follow carefully designed protocols to achieve overall good system performance. One effect of such cooperative caching is that some nodes have to possibly cache data items which are most accessed by other nodes instead of themselves. Such *mistreatment* [12] renders those nodes to break away from the group and operate in isolation using a local greedy replication scheme, in which they store the data they access most in the local memory, instead of data items most accessed by other nodes. It is important to take into consideration such selfish behaviors when we want to improve the system performance.

Several research, among many others, has been performed to address above selfish data caching behavior using game-theoretic analysis [2, 3, 7, 10]. Historically, data caching and the related cache placement problem have the theoretical root in facility location problem [6] and k-median problem [1], wherein facility is considered as cache node in the network. Both problems study how to place facility in the network with least cost to satisfy the access demands from the client nodes. Here the facility could be a delivery center, a distribution center, a transportation hub, or a restaurant. In the facility location problem, setting up a facility at a node incurs a certain fixed cost, and the goal is to minimize the sum of total access cost and the facility setting-up costs of all facilities, without any constraint. The k-median problem minimizes the total access cost under the number constraint, i.e., that at most k nodes can be selected as facilities, without considering setting-up costs. In both problems, the facilities to be set up are not initially in the network. As a result, most of work of selfish data caching assume that i) the data items are initially outside of the network and ii) the caching cost is either a constant or does not exist.

While this assumption is valid in many situations, there are applications where the data items are instead in the network and caching cost depends on where the data is located inside the network. In such applications, the caching cost depends on the network topology and distances among nodes. For example, in P2P networks, each peer initially has some data objects and shares them with other peers; in sensor networks, sensor nodes sense and generate data which are transmitted back to the base station for analysis or accessed by other sensor nodes in the network. In both cases, data are originally generated and stored in the network. Our model is geared towards multi-hop wireless ad hoc network, where the data items are initially at some nodes (called *source nodes*) in the network, and are subsequently cached by other nodes. The efficiency of data caching scheme in this paper depends on not only the network topology and nodes' access patterns, but also the data items' locations in the network.

In particular, in our network model, there is one data item contained in a single source node, and there are multiple client nodes that wish to access the data item. Different nodes have different demands (or access frequencies) towards the data item. The cache node caches the data from the closest existing cache node. Nodes that do not cache the data access the data from the closest cache node. The goal of the data caching problem is to determine a set of nodes in the network to cache the given data item, such that the total communication cost incurred in caching the data item and accessing the data item is minimized.

We first show that our problem is NP-hard. We then show a pure Nash Equilibrium exists in our data caching model via a centralized construction. In game theory, the Nash Equilibrium is the set of strategies taken by all the players such that no player will improve its benefit by changing its strategy unilaterally [14]. In our case, the caching strategies by each node is that they choose whether to cache or not the content so as to minimize their own cost. However, Nash Equilibrium may not guarantee system-wide performance – due to the selfishness of self-interested nodes, the resultant social cost (total cost) could be much larger than the social optimal cost (minimum total cost). Papadimitriou et al. [9, 16] illustrate this using *price of anarchy*, which is the ratio of the social cost of the worst possible Nash equilibrium to the cost of the social optimal solution. We show that in our constructed Nash Equilibrium, the price of anarchy could be as large as $O(N)$, where N is number of nodes in the network. Then, using an external incentive mechanism based upon a payment model, we show that social optimal can still be achieved while it is also a Nash Equilibrium (i.e., with the price of anarchy $O(1)$).

Paper Organization. The rest of the paper is organized as follows. Section II reviews both cooperative and selfish data caching in ad hoc networks and in distributed system as a whole. In Section III, we introduce our network model and formulate the data caching problem, and show its NP-hardness. In Section IV we formalize the selfish data caching problem and demonstrate by a construction that a pure Nash Equilibrium exists. Section V presents our payment model which achieves the optimal social cost as well as a Nash Equilibrium. In Section VI, we conclude the paper and point out some future work.

II. Related Work

A. Cooperative Caching in Ad Hoc Networks

There are lot of research designing distributed caching algorithms in ad hoc networks. Hara and Madria [5] are among the first to propose replica allocation methods in ad hoc networks, by taking into account the access frequency from mobile hosts to each data item and the status of the network connection. Yin and Cao [20] design and evaluate three simple distributed caching techniques, viz., *CacheData* which caches the passing-by data item, *CachePath* which caches the path to the nearest cache of the passing-by data item, and *HybridCache* which caches the data item if its size is small enough, else caches the path to the data. Fiore et al. [4] design a cooperative caching scheme to create a content diversity in ad hoc networks, so that a requesting user likely finds the desired information nearby. Zhao et al. [21] propose a novel asymmetric cooperative cache approach, where the data requests are transmitted to the cache layer on every node, but the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data.

Ko and Rubenstein [8] propose a distributed protocol that palaces replicated resources in a network such that the distance between identical copies of the same resource is large and each node is "close" to some copy of any resource. They study it by coloring each node, where each color is a replica the node is assigned. It proves the network can converge to a stable state following such protocol. In our previous work [18], we present a polynomial-time centralized approximation algorithm to replicate data, which reduces the total data access delay at least half of that obtained from the optimal solution. We also show a distributed caching technique derived from the centralized approximation algorithm.

The data caching we study in this paper is closely related to the rent-or-buy problem [17], a special case of the connected facility location problem [13, 15, 17]. Swamy et al. [17] give a 5-approximation algorithm. Nuggehalli et al. [15] study the same problem in the context of energy-efficient caching strategies in ad hoc networks. They provide a distributed solution that is within a factor of 6 of the optimal solution.

However, all of above work do not take into consideration of selfishness of the network node, which is the topic of this work.

B. Selfish Caching in Ad Hoc Networks and Distributed Systems

Chun et al. [2] are among the first to propose to study selfish caching in distributed systems using a game-theoretic approach. They consider one data object which is outside of the network. When a node decides to cache the data object, it assumes that the node always gets this data from outside of the network, which incurs a constant caching cost. A node either caches the data object in its local memory or accesses it from another node storing the object, depending on which costs less. They show that there exists a pure strategy Nash Equilibrium based on above model. However, the total social cost can not achieve optimum due to selfish behavior of players. They propose a payment model, in which each node bids for having an object replicated at another node, and show that both social optimal and Nash Equilibrium can be achieved.

Laoutaris et al. [10] study distributed selfish replication and caching of multiple objects. In their model, the set of objects are also not in the network initially and the caching cost is not considered. Moreover, the distances between nodes are not factored in when playing the game. Rather, it assumes that for each node, accessing an object from its local cache always costs t_l, from another cache node t_r, and from the origin server always t_s, with $t_l \leq t_r \leq t_s$. The contributions of their paper are two fold: a) they consider memory capacity of each node since multiple objects are involved; b) the Nash Equilibrium object placement strategies are implemented in a distributed manner. The authors further extend their work by identifying and and investigating the causes of mistreatment [11, 12].

Our work considers one in-network data item in a multi-hop ad hoc network, wherein both accessing cost and caching cost not only exist, but also are topology dependent.

III. Network Model and Problem Formulation

Network Model and Notations. We model the ad hoc network as a connected general graph, $G(V, E)$, where $V = \{1, 2, 3, ...n = |V|\}$ is set of nodes and E is set of edges. Two nodes are connected by an edge if they are within the transmission range of each other and thus can communicate directly. There is a single data item D in the network, which is stored in its original source node $S \in V$. D is requested by the nodes in the network – each node has its own access frequency towards the data; the access frequency of node $i \in V$ is $a_i \in R^+$. Let d_{ij} be the shortest distance (in terms of number of hops) between node i and node j, and let $d(i, M) = \min_{j \in M} d_{ij}$ be the shortest distance from i to some node in a set of nodes M. For a set of nodes $X \subseteq V$, its metric closure is defined as the complete graph upon X, wherein each edge is a shortest path between two nodes in X in the original network graph $G(V, E)$. Let $mst\_mc(X)$ denote the cost of a minimum spanning tree of the metric closure upon X. Given a set of cache nodes M where D is cached, the *caching cost* of node $i \in M$ is proportional to its distance to another cache node in M from which it caches the data; the *access cost* of a non-cache node i accessing D is $a_i d(i, M)$.

Let $\tau(M)$ denote the *total cost in the network* with a set of cache nodes M (source node $S \in M$), we have:

$$\tau(M) = \sum_{i \in V} a_i \times d(i, M) + \gamma \times mst\_mc(M) \quad (1)$$

In above equation, the two terms on the right hand side represent total access cost and total caching cost in the network respectively. Here γ is a constant that indicates the relative weight of caching cost compared to access cost. Notice that, by varying γ, we can model different network scenarios and requirements.

Data Caching Problem. The data caching problem based on above network model, which we call *in_Caching problem* in the rest of the paper, can be formulated as follows. Given a network graph $G(V, E)$, one data item D and its source node S, and access frequencies of all client nodes, the objective is to select a set of cache nodes $M \subseteq V$, such that the total cost in the network given by Equation 1 is minimized, i.e.,

$$\min_{M} \tau(M) \quad (2)$$

Let τ^{opt} denote the optimal cost, $\tau^{opt} = \min_M \tau(M)$. Let C^{opt} denote the set of cache nodes (including S) in the optimal solution, $C^{opt} = \text{argmin}_M \tau(M)$.

Difference Between Our Model and Rent-or-Buy Based Model. Our problem is closely related to the well-known rent-or-buy problem [17]. In rent-or-buy problem, one facility is already open, along with a set of locations at which facilities can be further built. Connecting the facilities incurs a cost which is proportional to the weight of the Steiner tree [19] connecting all the facilities, and each client accesses its closest facility. The objective is to find a solution (i.e., to select the locations to build facilities and connect them by a Steiner tree) which minimizes the total access cost and connecting cost. The rent-or-buy problem is known to be NP-hard [17].

In the rent-or-buy based data caching model, an intermediate node (called Steiner node) between two cache nodes (called terminal nodes) can also cache a copy of the data. In our model, however, a cache node always caches the data from another already existing cache node in the network, following a shortest path between these two nodes (and we call such shortest path the *caching path*). Therefore, by specifying which cache node caches data from which existing cache node, our model mandates the timeliness of selecting the cache nodes, which is the main characteristic distinguishing our data caching model from rent-or-buy based one.

Next, we show that the data caching problem defined above is NP-hard.

Theorem 1: The in_Caching problem is NP-hard.

Proof: The in_Caching problem can be proved to be NP-hard via a reduction from the facility location problem (FLP) [6]. The FLP is similar to in_Caching problem with two differences: i) the data item is initially outside of the network and ii) the caching cost is a constant. In in_caching problem, the caching cost of a cache node depends on its distance to another cache node from which it caches the data, thus is not the same for all the cache nodes. Therefore FLP is a special case of in_Caching when caching cost is a constant, which shows in_Caching problem is also NP-hard. ∎

IV. Selfish Caching Game in Multi-hop Ad Hoc Networks

In selfish caching game, however, whether a node caches the data itself or accesses the data from other cache nodes depends on which costs less, not on optimal solution. Below we first discuss the cost model in the selfish caching game, which is different from that in the data caching problem discussed in Section III. Then we show that a pure Nash Equilibrium exists. Finally we discuss the performance of Nash Equilibrium in terms of the price of anarchy, which quantifies the cost of the lack of cooperation among nodes.

A. Cost Model

For each non-source node, still, it either caches the data in its local memory (*cache node*) or accesses the data from others (*non-cache node*). There are two kinds of cost for node i: the *access cost* and the *caching cost*, which are denoted as α_i or β_i respectively.

Access Cost. For a non-cache node i, once all the cache nodes are selected and have data cached in their local memories, i accesses D from its closest cache node (including the source node). Assuming k is i's closest cache node storing D, then i's access cost α_i is $a_i d_{ik}$.

Caching Cost. When a cache node i decides to cache data from other existing cache nodes, it goes to the closest one, say k, to fetch the data and cache it into its local memory. Thus i's caching cost β_i is also proportional to the shortest distance to cache node k and $\beta_i = \gamma d_{ik}$. As in Section III, γ is a constant indicating the relative weight of caching cost to access cost.

Total Cost in Nash Equilibrium. In a caching game where each node is selfish, whether a node i is a cache node or not only depends on a_i and γ: if $a_i \geq \gamma$, i is a cache node and caches the data from its nearest existing cache node; if $a_i < \gamma$, it is a non-cache node and accesses the data from its nearest cache node. At Nash Equilibrium, each node is either a cache node or non-cache node. Let the *cost of node* i of requesting D be τ_i, then τ_i equals either α_i or β_i. Let τ^N denote the total cost of the network in Nash Equilibrium, $\tau^N = \sum_{i \in V} \tau_i$. Let C^N denote the set of cache nodes (including S) in a Nash Equilibrium as, $C^N = \{i | a_i \geq \gamma, 1 \leq i \leq n\} \cup \{S\}$. Let $m = |C^N|$.

Caching Strategy. The caching strategy of node i, denoted as C_i, includes the following. First, it decides whether it is a cache node or not. Second, if yes, it decides from which cache node (called its *parent cache node*) it fetches the data and caches in its local memory; if no, it decides from which cache node it accesses the data item D. More formally, $C_i = (n_i, p_i, c_i)$, where $n_i \in \{yes, no\}$ indicates if i is a cache or not, p_i is the parent cache node of i if i is a cache node, otherwise i accesses D from its closest cache node c_i. Let SP denote the *strategy profile* of the game, $SP = \{C_1, C_2, ..., C_n\}$. Thus SP shows the global cache placement and data access in the entire network.

Before we present our algorithm that achieves Nash Equilibrium, we first show a property of the Nash Equilibirum achieved in our caching game, which says that the cache node in Nash Equilibrium is still a cache node in the optimal solution.

Lemma 1: $C^N \subseteq C^{opt}$.

Proof: We prove it by contradiction. Assume node $i \in C^N$ is not a cache node in the optimal solution, i.e., $i \notin C^{opt}$. We have that $a_i \geq \gamma$. In an optimal solution (there could be multiple optimal solutions), assume i accesses another cache node l for the data item D. To further reduce the total cost, i can cache the data from l. This further reduces the total cost of the entire network by $(a_i - \gamma) \times d_{il}$, contradicting that it is an optimal solution. Therefore $C^N \subseteq C^{opt}$. ∎

We use C^A to denote the set of non-cache nodes in Nash Equilibrium that are cache nodes in the optimal solution, i.e., $C^A = C^{opt} - C^N$.

B. Nash Equilibrium Construction

Below, we first present the algorithm leading to a Nash Equilibrium. Then we present some observation from the algorithm, which serves as the basis of Nash Equilibrium proof in Theorem 2 later. Finally we discuss the Price of Anarchy of the achieved Nash Equilibrium. For the clarity of the presentation, we call a cache node before it caches data a *potential cache node*.

Nash Equilibrium Construction. The algorithm takes place in iterations. In each iteration, a node (potential cache node) is selected as cache node and caches data from an already existing cache node, the caching path being a shortest path between these two nodes. There are $m - 1$ non-source cache nodes, so the algorithm stops after $m - 1$ iterations.

Minimum Caching Cost Algorithm for Nash Equilibrium.

1) Start with the source node S, find a potential cache node which has the minimum shortest distance to the source node (where there is a tie, the potential cache node with smaller ID is selected). It is the new cache node. Move a copy of the data from the source node to the new cache node along the shortest path between them.
2) **while** (There is still un-cached potential cache node) Among all the shortest path linking any existing cache node (parent cache) to any potential cache node (child cache), find the one with the minimum cost. When there is a tie of the parent cache and/or child cache, choose the one with smaller ID. Move a copy of the data from the selected parent cache node to the selected child cache node along the shortest path between them.
3) For each of the non-cache nodes, find its nearest cache node from which it accesses the data.

Above is essentially the minimum spanning tree algorithm upon the metric closure of all the cache nodes, which is in the same line as the total caching cost modeled in Section III. Figure 1 (a) shows such Nash Equilibrium construction for a grid-like network topology, where each node can only communicate directly with its (at most four) neighbors. All the caching paths are shown as the arrowed edges, the direction of which indicating the movement of

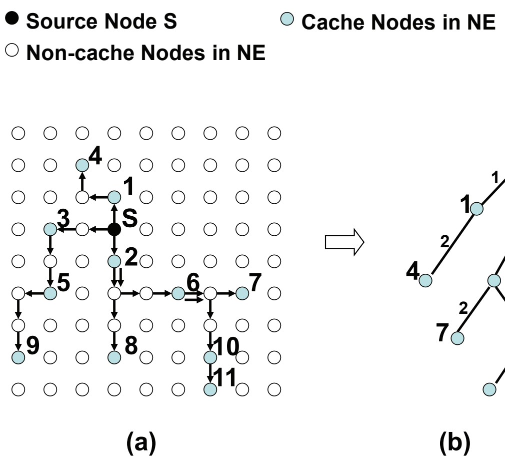

Fig. 1. Nash Equilibrium construction in a grid-like ad hoc network. (a) shows the caching paths. The ID of a cache node is the time sequence at which it gets a data copy from its parent cache following the direction of the arrowed edge. Note that some edges are used multiple times, indicated by the double arrowed edges. (b) shows the cache tree in the Nash Equilibrium, with vertices as the cache nodes and the number on the edge the cost of the shortest path between each pair of parent-child cache nodes.

the data item from parent cache node to child cache node. Note that the grid-like topology is only for the purpose of ease of presentation, above Nash Equilibrium construction is applicable to any topologies. For the ease of presentation, the ID of each node in the figure indicates the iteration (time sequence) at which the cache node gets a data copy from its parent cache following the caching path, not its actually ID.

Above algorithm gives the parent-child relationship of all the cache nodes, indicating from which parent cache node that each child cache node directly caches the data item from. Consequently, this forms a tree rooted at source node S, which we call the *cache tree*.

Cache Tree. Figure 1 (b) shows the cache tree corresponding to the caching paths in Figure 1 (a). The vertices of the cache tree is the set of cache nodes. Each edge represents a parent-child cache node relationship. The number on the edge indicates the cost of the shortest path between each pair of parent-child cache nodes (recall that the caching cost is proportional to such shortest path cost). Note that the cache tree is a "logical" tree, because its corresponding caching paths do not necessarily form a tree, as shown in Figure 1 (a). Below we give some definitions related to the cache tree.

Definition 1: (Parent Cache and Child Cache.) In cache tree, cache node i is the *parent cache* of cache node j, denoted as $P(i)$, if j directly fetches the data from i. j is a *child cache* of i. $\square$

Definition 2: (Ancestor Cache and Descendant Cache.) Cache node i is an *ancestor cache* of cache node j ($i \neq j$) if i is on the unique path from S to j in the cache tree. That is, j directly or indirectly fetches the data from i. j is a *descendant cache* of i. $\square$

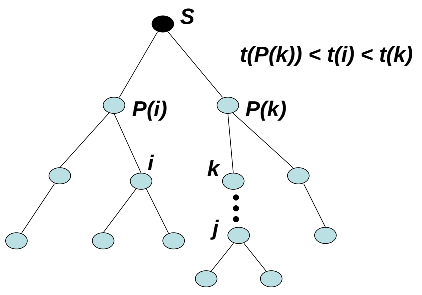

Fig. 2. In a cache tree, if $t(j) > t(i)$ and $j \notin D(i)$, then $|iP(i)| \leq |ij|$, which means i does not have incentive to deviate from $P(i)$ to have cache node j as its parent cache.

Definition 3: (Descendant Set.) The *descendant set* of cache node i, denoted as $D(i)$, is all the nodes in the subtree rooted at i. That is, such subtree is the set of i's descendant caches. For example, the descendant set $D(S)$ of source node S is the whole cache tree excluding S itself. $\square$

Definition 4: (Selected Time of Cache Node.) Since in caching game, each potential cache node is selected as cache node one by one in a sequence and caches data, the cache nodes in the resulted cache tree can be ordered using their selected time. We use $t(i)$ to indicate the time sequence at which i is selected as cache node and assume $t(S) = 0$. For any cache node i, we have $t(D(i)) > t(i)$. $\square$

Discussion of the Selection of Parent Cache. In the caching game, when i is selected to become a cache node, it chooses the nearest *existing* cache node, $P(i)$, as its parent cache and fetches the data from $P(i)$. However, a later selected cache node j could be closer to i than $P(i)$, causing cache node i to deviate and cache the data from j instead. Surprisingly, we show that for all the cache nodes selected after i, only nodes in $D(i)$ can possibly be closer to i than $P(i)$. Formally, Lemma 2 below shows that if a cache node j caches after cache node i, and j is not a descendant cache of i, then i does not have incentive to deviate to cache from j.

Lemma 2: For two cache nodes i and j in a cache tree, if $t(j) > t(i)$ and $j \notin D(i)$, then i does not have incentive to deviate from its parent cache $P(i)$ to have j as its parent cache.

Proof: As shown in Figure 2, since $t(j) > t(i)$ and $j \notin D(i)$, along the path from j to S (including j and S), there must exist one cache node, say k, with $t(P(k)) < t(i) < t(k)$ (note $P(k)$ and $P(i)$ could be the same node). Denoting the cost of the shortest distance between cache nodes A and B as $|AB|$, we have $|iP(i)| \leq |kP(k)|$. This is because at the iteration when i is selected as the cache node and caches data from $P(i)$, $iP(i)$ is the minimum shortest path among all the shortest paths connecting any cache node to any potential cache node. Using similar argument, we have $|kP(k)| \leq |ji|$. So we have $|iP(i)| \leq |ij|$, i does not have incentive to cache from j. ∎

For example, in Figure 1 (a), since $t(6) < t(9)$ and node $9 \notin D(6)$, node 6 will not deviate from its parent cache node 2 to cache from node 9. This can be confirmed by that the distance between node 6 and node 2 is 3 hops, which is less than distance between node 6 and node 9, which is 7 hops.

Assumption. In our in-network data caching model, we assume that an ancestor node can not access or cache data from its descendant cache nodes. We believe this is a valid assumption because otherwise, it violates the intrinsic ancestor-descendant relationship between ancestor and descendant nodes. This assumption, together with Lemma 2, lead to below theorem about the Nash Equilibrium construction.

Theorem 2: The minimum caching cost algorithm reaches Nash Equilibrium.

Proof: To prove that Nash Equilibrium exists, we need to show each node does not unilaterally deviate from its caching strategy. That is, the caching node keeps its parent cache node, while non-cache node accesses data from the same cache node. For any non-cache node i, it will not deviate to be a caching node since its access cost is less than its caching cost, due to $a_i < \gamma$; it will not access from another cache node since it accesses the data from closest cache node.

For any cache node i, it will not deviate to be a non-cache node because $a_i \geq \gamma$. Below we show it will not deviate from its parent cache node $P(i)$ to any other cache node, say j. This is true when $t(i) > t(j)$, since $|iP(i)| \leq |ij|$ following the minimum caching cost algorithm. When $t(i) < t(j)$ and $j \notin D(i)$, from Lemma 2 i will not deviate to have j as its parent cache node. When $t(i) < t(j)$ and $j \in D(i)$, it does not deviate either since it is prohibited that ancestor node accesses or caches data from its descendant cache.

No node deviates unilaterally. We conclude that the minimum caching cost algorithm reaches Nash Equilibrium. ∎

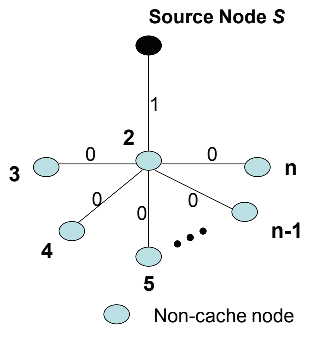

Fig. 3. An example showing $PoA = O(n)$ in the selfish caching game.

C. Price of Anarchy (PoA)

Price of Anarchy is defined as the ratio between the social cost of the worst possible Nash equilibrium to the cost of the social optimal solution [9, 16]. Below we discuss the PoA in our constructed Nash Equilibrium.

Lemma 3: If $a_i \geq \gamma$ for all $i \in V$, the PoA of the data caching game is O(1).

Proof: In this case, all the nodes in the network are cache nodes, and the total cost is $\tau = \sum_{i \in V} \gamma d_{iP(i)}$. The metric closure upon all the cache nodes is the same as $G(V, E)$. The minimum caching cost algorithm upon the metric closure is essentially the same as the minimum spanning tree algorithm upon $G(V, E)$. Thus both yield the same cost, PoA of the game is $O(1)$. ∎

Lemma 4: If $\exists\ i \in V$, such that $a_i < \gamma$, the PoA of the game can be $O(n)$.

Proof: We prove this by showing an example depicted in Figure 3. In this example, nodes $\{2, 3, ..., n-1, n\}$ are all non-cache nodes, with $a_2 = a_3 = a_n = a < \gamma$. The distance between node 2 and source node S is 1 and the distance between nodes $3, 4, ..., n-1, n$ to node 2 is 0. The Nash Equilibrium is that all the non-cache nodes access the data from S, giving total cost $\tau^N = a \times (n-1)$. However, the social optimal solution is that node 2 caches the data while nodes $3, 4, ..., n-1, n$ access it from node 2, yielding optimal total cost $\tau^{opt} = \gamma$. Therefore PoA $= \frac{\tau^N}{\tau^{opt}} = \frac{(n-1)a}{\gamma}$, which is $O(n)$. ∎

Lemma 4 shows that due to the non-cooperation among selfish nodes, the social optimal is not achieved in the Nash Equilibrium. Below, we present a payment-based mechanism wherein some non-cache nodes are made payment by other nodes. We show our proposed mechanism can achieve both Nash Equilibrium and social optimal.

V. Payment Mechanism

In this section, we design a payment-based mechanism wherein a Nash Equilibrium is achieved while its total cost is equal to the social optimal. Note C^A is the set of non-cache nodes in original Nash Equilibrium that are cache nodes in the optimal solution. The idea of our payment-based mechanism is to motivate the nodes in C^A to cache data such that the total cost of the network is optimal. For this, each node who benefits from such caching makes some amount of bid to this node. If this node caches the data, then the bidding node must pay the bided amount to the caching node. The payment mechanism also decides for each caching node in C^A, beyond how much bid it receives that it is willing to cache the data. Using this payment mechanism, we show that both Nash Equilibrium and social optimal can be achieved.

As in [2], we define the strategy of each node i in the payment game as a triplet $(v_i, b_i, t_i) \in \{N, R+, R+\}$, indicating i) node i makes b_i amount of bid to node v_i and ii) node i's threshold value of received bid is t_i beyond which i will cache the data. We use B_i to denote the total amount of bid that node i receives, i.e., $B_i = \sum_{\{j|i=v_j\}} b_j$. A node i will cache the data if and only if $B_i \geq t_i$. Below we first present some properties of the cache tree resulted from the optimal solutions.

Lemma 5: For two cache nodes $i, j \in C^{opt}$, if $i \in C^A$ and $j \notin D(i)$, i.e., $j \in \{C^{opt} - \{i\} \cup D(i)\}$, then $P(j)$ is the same node in either the basic Nash Equilibrium or the optimal solutions. j's caching cost does not depend on whether i caches or not. In other words, removal of i from C^{opt} will not affect the caching path of $j \notin D(i)$.

Proof: If $t(j) < t(i)$, j's caching path is not affected because it caches before i caches. If $t(j) > t(i)$, since $j \notin D(i)$, by way of contradiction, if j changes its parent cache from $P(j)$ to another cache node, say k, as the result of minimum spanning tree algorithm upon the metric closure of all the nodes in $C^{opt} - \{i\}$, this results that the cache tree of all nodes in C^{opt} is not minimum, which contradicts with the fact that such tree is minimum spanning tree upon the metric closure of all the nodes in C^{opt}. ∎

So only nodes in $D(i)$ can possibly change their caching paths if $i \in C^A$ decides not to cache. Now we discuss the payment mechanism in the optimal solution. For each cache node $i \in C^A$, we denote the set of non-cache nodes accessing data from i as Θ_i.

Definition 5: (Benefit of cache node $i \in C^A$ to non-cache node $j \in \Theta_i$.) We define the benefit of cache node i to non-cache node $j \in \Theta_i$ (i.e., $a_j < \gamma$), denoted as ψ_j, as the minimum extra cost incurred to j if i is not a cache, and therefore j has to *access* the data from the closest cache node in $C^{opt} - \{i\}$. Formally, $\psi_j = a_j \times (min_{k \in (C^{opt} - \{i\})} d_{jk} - d_{ji})$. □

Definition 6: (Benefit of cache node $i \in C^A$ to another cache node $k \in D(i)$.) We define the benefit of cache node i to another cache node $k \in D(i)$, denoted as ϕ_k, as the minimum extra cost incurred to k if i decides not to cache, thus k has to *cache* the data from another cache node in $C^{opt} - \{i\}$. Formally, $\phi_k = \gamma \times (d_{P'(k)k} - d_{P(k)k})$, where $P'(k)$ is the parent cache of k following the minimum spanning tree algorithm over the metric closure of the set of cache nodes are $C^{opt} - \{i\}$, and $P(k)$ is the parent cache of k when the set of cache nodes are C^{opt}. □

Definition 7: (Benefit and average benefit of cache node $i \in C^A$ to the entire network.) Now, the benefit of cache node i to the entire network, denoted as NB_i, is the saving of the total cost due to caching at i. Formally, $NB_i = \sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k - (\gamma - a_i)$. The average benefit of node i, denoted as nb_i, is the average saving for all the nodes in Θ_i and $D(i)$, plus itself, if node i decides to cache. That it, $nb_i = NB_i/(|\Theta_i| + |D(i)| + 1)$. □

Lemma 6: $nb_i \geq 0$.

Proof: By way of contradiction, assume $nb_i < 0$ in the optimal solution. That is, $\sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k - (\gamma - a_i) < 0$. Consider a new caching solution where i decides not to cache and thus each node in Θ_i and D(i) chooses its next best strategy. All other nodes in $C^{opt} - i$ are still cache nodes. It is easy to see that the optimal cost minus the cost of the new caching solution is at least:

$$\begin{aligned}
& (\gamma + \sum_{j \in \Theta_i} d_{ji} + \sum_{k \in D(i)} d_{P(k)k}) - \\
& (a_i + \sum_{j \in \Theta_i} min_{k \in (C^{opt} - \{i\})} d_{jk} + \sum_{k \in D(i)} d_{P'(k)k}) \\
= \; & (\gamma - a_i) - \big(\sum_{j \in \Theta_i} min_{k \in (C^{opt} - \{i\})} d_{jk} - d_{ji} + \\
& \sum_{k \in D(i)} (d_{P'(k)k} - d_{P(k)k}) \big) \\
= \; & (\gamma - a_i) - (\sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k) \\
> \; & 0,
\end{aligned}$$

which contradicts the optimality of the optimal solution. ∎

Payment mechanism. The bid of each node is set as follows. For each $j \in \Theta_i$, the amount j bids i is $b_j = max\{0, \psi_j - nb_i\}$. For each $k \in D(i)$, the amount k bids i is $b_k = max\{0, \phi_k - nb_i\}$. That is, each node bids the amount which it benefits more than the average benefit of the network due to i's caching. For other nodes $l \notin \Theta_i \cup D(i) \cup \{i\}$, the amount l bids i is $b_l = 0$.

The threshold of i t_i is given as:

$$t_i = \begin{cases} 0 & \text{if } i \in C^N \\ \sum_{j \in \Theta_i} b_j + \sum_{k \in D(i)} b_k & \text{if } i \in C^A \end{cases}$$

For all the non-cache nodes in the optimal solution, their threshold is 0 too.

Below we show that above payment mechanism yields social optimal as well as Nash Equilibrium.

Theorem 3: The payment mechanism reaches Nash Equilibrium, and it yields social optimal for the entire network.

Proof: We need to show with the payment mechanism, all the nodes in the optimal solution has no incentive to deviate, as long as others stay with their strategies.

First, we show that for node $i \in C^A$, it better off caches the data in spite of the fact that $a_i \leq \gamma$. We have

$$\begin{aligned}
B_i &= \sum_{j \in \Theta_i} b_j + \sum_{k \in D(i)} b_k \\
&\geq \sum_{j \in \Theta_i} (\psi_j - nb_i) + \sum_{k \in D(i)} (\phi_k - nb_i) \\
&= \sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k - (\sum_{j \in \Theta_i} + \sum_{k \in D(i)}) \times nb_i \\
&\geq \sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k \\
&\quad -(|\Theta_i| + |D(i)|) \times NB_i/(|\Theta_i| + |D(i)| + 1) \\
&\geq \sum_{j \in \Theta_i} \psi_j + \sum_{k \in D(i)} \phi_k \\
&\quad -NB_i + NB_i/(|\Theta_i| + |D(i)| + 1) \\
&\geq \gamma - a_i + nb_i \qquad \text{Definition of } nb_i \\
&\geq \gamma - a_i \qquad \text{From Lemma 6}
\end{aligned}$$

Above shows that for cache nodes in the optimal solution that are not cache node in the previous Nash Equilibrium, the amount of bids they collect is more than the extra cost they incur due to caching. They better off to caching and has no incentive to deviate.

∎

VI. Conclusion and Future Work

We apply game-theoretical analysis for the selfish caching in wireless ad hoc networks. Our model considers distance-dependent caching cost, which is different from previous work. We first show a pure Nash Equilibrium exists in our model. We then design a payment model, in which the selfish caching game achieves both optimal cost and Nash Equilibrium simultaneously. Our model is more general and applicable than existing work for such emerging networks as P2P and wireless ad hoc sensor networks. As the ongoing and future work, we are validating our findings using simulations under various network scenarios.

References

[1] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proc. of IEEE FOCS 1999*.

[2] B.-G. Chun, K.Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou, and J. Kubiatowicz. Selfish caching in distributed systems: A game-theoretic analysis. In *Proc. of ACM PODC 2004*, pages 21 – 30.

[3] O. Ercetin and L. Tassiulas. Market-based resource allocation for content delivery in the internet. *IEEE Transactions on Computers*, 52(12):1573–1585, 2003.

[4] Marco Fiore, Francesco Mininni, Claudio Casetti, and Carla-Fabiana Chiasserini. To cache or not to cache. In *Proc. of IEEE INFOCOM 2009*.

[5] Takahiro Hara and Sanjay K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1515–1532, 2006.

[6] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[7] S. U. Khan and I. Ahmad. A pure nash equilibrium based game theoretical method for data replication across multiple servers. *IEEE Transactions on Knowledge and Data Engineering*, 20(3), 2009.

[8] B.-J. Ko and D. Rubenstein. Distributed self-stablizing placement of replicated resources in emerging networks. *IEEE/ACM Transactions on Networking*, 13(3):476 – 487, 2005.

[9] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proc. of STACS 1999*.

[10] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed selfish replication. *IEEE Transactions on Parallel and Distributed Systems*, 17:1401–1413, 2005.

[11] Nikolaos Laoutaris, Georgios Smaragdakis, Azer Bestavros, Ibrahim Matta, and Ioannis Stavrakakis. Distributed selfish caching. *IEEE Transactions on Parallel and Distributed Systems*, 18:1361–1376, 2007.

[12] Nikolaos Laoutaris, Georgios Smaragdakis, Azer Bestavros, and Ioannis Stavrakakis. Mistreatment in distributed caching groups - causes and implications. In *Proc. of IEEE INFOCOM 2006*.

[13] P. Mirchandani and R. Francis. Discrete location theory. 1990.

[14] John Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, 1951.

[15] Pavan Nuggehalli, Vikram Srinivasan, and Carla-Fabiana Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *Proc. of MOBIHOC 2003*.

[16] C. Papadimitriou. Mistreatment in distributed caching groups causes and implications. In *Proc. of ACM STOC 2001*.

[17] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. In *Proc. of APPROX 2002*.

[18] Bin Tang, Samir Das, and Himanshu Gupta. Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(3):289–304, 2008.

[19] P. Winter. Steiner problem in networks: A survey. *ACM Networks*, 17(2):129–167, 1987.

[20] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.

[21] Jing Zhao, Ping Zhang, Guohong Cao, and Chita R. Das. Cooperative caching in wireless p2p networks: Design, implementation, and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 99(2):1045–9219, 2009.