Revenue-Rewarding Scheme for Cache Provided in Proxy Caching System of Media Stream

ALAN TAK-SHUN IP
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
and
JIANGCHUAN LIU
Simon Fraser University, Vancouver, BC, Canada
and
JOHN C.S. LUI
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

Network entities cooperating together can improve system performance of media streaming. However, the lack of participation incentive prohibits the deployment of such cooperative systems. In this paper, we address the incentive issue of a cooperative proxy caching system. We study the problem of what motivates each proxy to provide cache space to the system. To encourage proxies to participate, we suggest a revenue-rewarding scheme to credit the cooperative proxies according to the resources they contribute. Game-theoretic model is used to analyze the interactions among proxies under the revenue-rewarding scheme. Since no system-wide property is achieved in the non-cooperative environment, we suggest two cooperative game settings that lead to socially optimal situations, where the benefits of the network entities are maximized. The evaluation shows with the incentive mechanism incorporated, the proxies have a strong incentive to collaborate in the system, and the optimal net profit and social welfare are achieved in the cooperative resource allocation games.

Categories and Subject Descriptors: []:

General Terms:
Additional Key Words and Phrases: Incentive, Resource allocation, Pricing, Game, Nash equilibrium

1. INTRODUCTION

Cooperative networks especially P2P networks have caught much attention in recent years. In such systems, network entities collaborate with each others by sharing their own resource, such as storage, bandwidth or computational power, to form a resource pool, and this aggregated resource pool helps to improve the system performance. Many real applications have been deployed, such as distributed file sharing [Kazaa] [BitTorrent], collaborative web caching, P2P streaming [Zhang et al. 2005], distributed computing, etc. It is generally agreed that the cooperative network performs significantly better than the traditional client-server model in supporting large amount of users. In short, it provides an inexpensive platform for application that requires scalability, efficiency and robustness.

However, most cooperative systems assume that peers are "voluntary" to contribute. In fact, this assumption is not realistic. The autonomous peers are selfish in nature, and without concrete incentive, there is no motivation to contribute resources, by which they incur service degradation or suffer from cost. A study in Gnutella file sharing system [Adar and Huberman 2000] suggested that over 70% of users share little or no content. The large-scale deployment of the cooperative systems are obstructed by the free-riding problem, and motivating the users to cooperate is critical to the success of such systems.

To increase the involvement of peers, participation incentive mechanisms [Lui et al. 2002] have to be used to effectively encourage the users' collaboration in the network [Ranganathan et al. 2003]. Different approaches have been proposed in the literature. Better quality of service is given to the peers who contribute to the network, while free-riders are discriminated against. However, effective resource allocation that differentiates the contributors in a highly dynamic network is complicated. Others suggested using the reputation based system, where reputations of the participating peers are accumulated so as to reflect their contribution. The major issue here is how to quantify the user's contribution. Also, a secure and trusted reputation system is essential to prevent fake reputation, but it is difficult to achieve without a centralized authority. Nevertheless, whitewashing is possible for the malicious user by pretending to be another user.

Another approach is to setup a contribution-rewarding mechanism to credit the peers cooperating in the system. The reward may come from the overall revenue of the cooperative network, by means of service pricing or cost reduction. The simplest way to achieve this goal is to grant a fixed credit to a peer whenever it participates. Such a scheme can be implemented easily, but it is unfair to the peer who contribute more resource. We can also reward the peers in proportional to the resources they contributed. This scheme not only achieves proportional fairness, but also encourages peers to supply sufficient amount of resource. By rewarding appropriately, sufficient amount of resources are supplied by the peers, and the efficiency of the overall system is improved.

In this paper, we propose a revenue-rewarding mechanism to encourage cache supplying in a cooperative proxy caching system for media streaming. [Ip et al. 2005] showed that proxies cooperating together can significantly reduce the aggregated transmission cost in delivering the video streams, and it is worthwhile to setup the cooperative system. However, they did not address the incentive issue. Our revenue-rewarding scheme works complementary with the cooperative network in stimulating participation from the proxies. In fact, a cost-profit analysis has suggested that it is profitable to setup an incentive-based cooperative system for media streaming [Mohamed M. Hefeeda and Bhargava 2003].

Our work follows the contribution reward-based incentive approach to reward the contributors by the partial revenue obtained from the cooperative system. We focus on how peers' contributions are influenced by the revenue-rewarding scheme. Game theoretic model is used to analyze the interaction between proxies under different resource allocation games. We show that in the non-cooperative environment, the proxies selfishly optimize its own utility. As a result, the best total benefit received by the network nodes are not guaranteed. We further propose two cooperative resource allocation games that lead to two different optimal situations. We examine the performance of the scheme in terms of profit maximization and utility maximization. By evaluating the net profit and the social welfare received by the network entities, we demonstrate that the proposed game settings motivate different entities in the network to cooperate. In addition, two system-wide objectives, net profit and social welfare, are achieved. Also, the resulted resource allocation is cost-effective as only the proxies with low cost participate in the system.

Our contributions are summarized as following:

(1) Proxies are encouraged to cooperate under the revenue-rewarding scheme.
(2) Net profit and social welfare are maximized in the cooperative games.
(3) Cost effective resource allocation is achieved in the cooperative games.

The remainder of the paper is organized as follows. Section 2 gives an overview of the system, and presents the mathematical formulation. Section 3 describes the revenue-rewarding scheme applied in the three resource allocation games: Non-cooperative game, Profit maximizing game and Utility maximizing game. The performance of these games are evaluated in Section 4. Section 5 reviews the related work. Finally, Section 6 concludes the paper.

2. SYSTEM MODEL

2.1 System Overview

We consider a cooperative proxy caching system for multimedia streaming. The architecture of this caching system is shown in Fig. 1. It consists of a logical video server, a number of proxies and their clients, and a network service provider (**NSP**). The NSP provides solely the network connection service to the entities in the network. The client requests for videos, which are streamed from the far-located server to the client through the intermediate proxies. The proxies are capable of caching the video stream passing through them. Each video is divided into equal-sized segments for caching, and whether a segment is being cached in the proxy is determined by the cache allocation algorithm. In general, the frequently accessed video segments are cached in the local proxy to reduce network traffic. The proxies are logically connected by direct or indirect links. They cooperate with each others by sharing the cached segments among themselves, i.e. a proxy can request for a video segment cached in other proxies.

This is the COPACC architecture proposed in [Ip et al. 2005], which is a cooperative proxy-and-client caching system. The COPACC system aims at reducing the aggregated transmission cost by allocating efficiently the video segments to the cache provided by the proxies and clients. According to the cache allocation algorithm, videos are partitioned into prefix (P^i), prefix-of-suffix (Q^i) and the remaining suffix, and the proxies and clients are responsible to cache the prefix and prefix-of-suffix respectively. Based on the video transmission scheme used(either unicast or multicast), the optimal partitioning of the videos are computed to minimize the aggregated transmission cost, i.e. the values of P^i and Q^i are determined

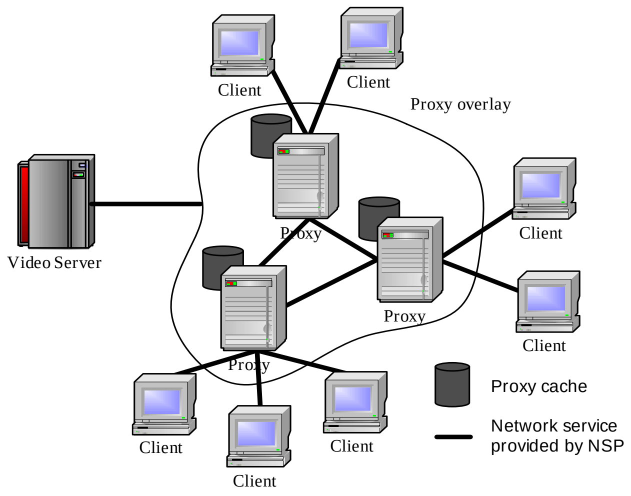


Fig. 1. The architecture of the cooperative proxy caching system.

to minimize $\sum_i Cost(P^i, Q^i)$. The optimal prefix and prefix-of-suffix are further divided into smaller segments in order to fit in multiple proxies and clients. Optimal placement of these segments into the proxies and clients is also considered to minimize the cost. The merit of the COPACC relies on the proxy cooperation. However, COPACC did not address the incentive issues for the proxy's participation. That is, what motivates each proxy to provide the cache space and how much cache space should be allocated. We extend COPACC by proposing a revenue-rewarding scheme to provide incentive for the proxies to cooperate.

In order to increase profit, the NSP is keen to admit new clients. However, since the capacity of the network links is limited, the NSP fails to serve a large number of video-streaming users having high bandwidth and short delay requirements. Unfortunately, upgrading the network facility is not desirable because the investment cost is usually high. A cost effective approach is to setup a COPACC system to reduce the aggregated transmission cost in the network. As such, the same link capacity can accomplish more clients. Hence, the NSP has a strong incentive to encourage the proxies to participate in the COPACC system.

In general, the more the cache, the better the performance of the system is. From the NSP's prospective, it wants more cache supplied because insufficient cache space results in a small cost reduction. However, resources are not supplied for free. The proxy has to pay certain cost to maintain the resources, although the cost is usually implicit. Therefore, the proxies participated in the incentive-based COPACC system have to decide carefully the amount of cache storage to be contributed. If they contribute too little storage, the reward is small; if they contribute too much storage, the cost of maintaining the cache is higher than the reward. It is assumed that the cost follows the general rule of increasing marginal cost, i.e. the cost of providing an additional unit of cache is higher than that of the previous unit. Thus, proxies are reluctant to provide too much resources to the system.

As the consequence, a revenue-rewarding scheme is established by the NSP to

Parameter	Definition
H	Number of proxies.
s_i	Cache space supplied by proxy i.
$\hat{S}_i$	Storage capacity of proxy i.
q	Total cache space supplied to the system.
$C_i(s_i)$	Cost of supplying s_i unit of cache space by proxy i.
$R(q)$	Revenue of the system with q units of cache space supplied.
$P(q)$	Credit granted to the proxy for each unit of cache space supplied.
$U_i(s_i)$	Utility of supplying s_i unit of cache space by proxy i.
$E(q)$	Net profit of the NSP in the system with q units of cache space supplied.
$SU(s_1, s_2, ..., s_H)$	Social Utility of the system.

Table I. A summary of the notations.

reward the contributing proxies. The reward, in terms of credit, is determined based on the amount of resource shared by the proxy. It is proportional to the proxy's contribution. Proxies are rewarded regularly for every fixed period of time. Only the proxies with full participation throughout the period are qualified for the rewards. This encourages the proxies to stay in the network until the end of each period, thus avoiding the unpredictable proxy leave in the system.

An authority, such as the NSP, is responsible to define a price value, which specifies how much credit per unit storage should be granted to the participating proxy. Ideally, the price should match the demand and supply of resource such that social optimal is achieved. However, it is not the case in a non-cooperative environment. Given that proxies are selfish in nature, they strategically allocate the amount of storage that maximizes their benefit only, i.e. maximize the reward minus cost, regardless of other proxies. Meanwhile, the NSP wants to achieve the largest benefit by giving out less reward. This forms a non-cooperative game between the NSP and the proxies that often leads to a non-optimal situation. In this case, the proxies tend to over-supply the resource.

2.2 System Formulation

We use a game-theoretic approach to model the economic of the resource supplying from the proxies. There are two kinds of player, the NSP and the proxy. The NSP provides network connectivity to the proxies, while the proxy provides cache storage to reduce the transmission cost of the system. The notations used in the paper are summarized in Table I.

There are H proxies cooperating in the system. Let s_i be the unit of cache space that proxy i decided to allocate to the system, and $\hat{S}_i$ be the maximum storage capacity of the proxy. A feasible s_i is the unit of cache space the proxy can supply, i.e. $0 \le s_i \le \hat{S}_i$. The sum of the cache space supplied to the system is $q = \sum_{i=1}^{H} s_i$. In order to supply s_i units of cache space, proxy i has to pay $C_i(s_i)$, where $C_i(s_i)$ is the cost function of supplying s_i from proxy i, and the cost function can be heterogeneous between different proxies. In this paper, we consider the cost function to be strictly increasing with convex shape. The cost function for proxy i is defined as following:

$$C_i(s_i) = \begin{cases} A_i e^{\theta_i (s_i - b_i)}, & 0 < s_i \le \hat{S}_i \\ 0, & s_i = 0. \end{cases} \tag{1}$$

We argue that the exponential cost function is suitable because it reflects the general rule of increasing marginal cost. The parameter A_i defines the initial cost of setting up the proxy, while θ_i determines the increasing rate of the cost. For example, a cost function with large value of A_i and θ_i have a high cost. When θ_i is set to zero, the cost function becomes a constant A_i, meaning that the cost is fixed regardless of the amount of cache supplied. Each proxy can assign its own cost function by adjusting the parameters A_i, θ_i and b_i.

The NSP is in charged to estimate the revenue $R(q)$ in the system. For example, the revenue function can be obtained from the COPACC system by approximating the transmission cost reduction with respect to the total cache space q. In general, the more the resources supplied by the peers, the higher the revenue. However, the marginal revenue is decreasing as the resource increased. When the cache space reaches a specific amount, the cost reduction approaches the limit. Thus, we model the revenue function as a non-decreasing and concave function, which is defined as

$$R(q) = \frac{A'}{\theta'}[1 - e^{-\theta'(q-b')}], \qquad q > 0. \tag{2}$$

The price is a function of q, and it is set according to the revenue curve. The product of the price and the total available cache unit q should not exceed the corresponding revenue. The NSP has its freedom to decide how much revenue is rewarded to the proxies, by setting an appropriate price function. We suggest two possible ways to define the price function.

(1) **Total-rewarded pricing**: The price function $P(q)$ is defined as the revenue divided by the total resource supplied, i.e. $P(q) = R(q)/q$ for $q > 0$.

(2) **Marginal-rewarded pricing**: The price function $P(q)$ is defined as the marginal gain of the system, i.e. $P(q) = R'(q)$.

In general, $P(q)$ is a decreasing function with a convex shape. When the amount of resource tends to infinite, the price of each unit of resource approaches to zero. The total and marginal-rewarding price are defined as following:

$$P(q) = \frac{A'}{\theta' q}[1 - e^{-\theta'(q-b')}], \qquad q > 0. \tag{3}$$

$$P(q) = A' e^{-\theta'(q-b')}, \qquad q > 0 \tag{4}$$

We like to emphasize that this methodology is not restricted to use in the caching system, but it may also be applied to other P2P system with the cost and the rewarding function setup properly. We now present the resource allocation game among the proxies in the COPACC.

3. RESOURCE ALLOCATION GAME

We model the behavior of the proxies and the NSP as a strategic game. All proxies (or system administrators who manage the proxy) are rational, and they strategically choose the amount of cache s_i to maximize their benefit. We use a utility function to represent the level of satisfaction of the proxy. The utility $U_i(s_i)$ of proxy i can be measured in terms of its net gain, which is equivalent to the reward earned minus the cost to provide the cache. The utility is expressed as follow:

$$U_i(s_i) = s_i P(q) - C_i(s_i). \tag{5}$$

The resource allocation game is a repeated asynchronous game. Each proxy can make or change its decision about the amount of cache at the beginning of each round. To be realistic and scalable, we assume imperfect knowledge of each proxy, meaning that the proxy only knows about the total cache space supplied to the system, q, and the price function, $P(q)$. The NSP (or proxy coordinator) can publicize the current price and the total amount of cache space such that other proxies can obtain the information easily. Based on these information, the proxy updates its own strategy in each move to maximize its utility.

3.1 Non-Cooperative Game

In the non-cooperative game, the proxies make decision regardless of the other proxies. They choose s_i based on the public information: the aggregated cache space and the price function. The objective of each proxy is to maximize its own utility with respect to s_i over $[0, \hat{S}_i]$:

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = s_i P(q) - C_i(s_i). \tag{6}$$

Given the total cache space q and the price function $P(q)$, the proxy can determine its best strategy s_i by solving the maximization problem. Note that q is implicitly depends on s_i. If the value of s_i is changed, the value of q, as well as $P(q)$, will be adjusted accordingly. Thus, in the optimization, the value of q would be better presented in terms of s_i. Let s_{-i} be the amount of cache collectively supplied by the proxies except proxy i, then $s_{-i} = q' - s'_i$, where q' and s'_i are the total amount of cache and the amount of cache supplied by proxy i respectively in the previous round. The equivalent optimization problem is shown as following:

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = s_i P(s_i + s_{-i}) - C_i(s_i). \tag{7}$$

Specifically, in the COPACC system, the objective function can be written as

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = \begin{cases} s_i A' e^{-\theta'(s_i + s_{-i} - b')} - A_i e^{\theta_i (s_i - b_i)}, & 0 < s_i \le \hat{S}_i \\ 0, & s_i = 0. \end{cases} \tag{8}$$

The marginal-rewarded pricing is used here. This maximization problem is simple, and the first-order condition is sufficient to solve the optimal value of s_i. In general, the game will converge to a Nash equilibrium. However, the Nash equilibrium may not be unique as the order of move will influence the equilibrium point. The first mover is more likely to get advantage over the later mover by supplying more cache space at the beginning. The outcome of this non-cooperative game is not desirable since there is no guarantee that the equilibrium is socially optimal.

In the non-cooperative environment, the proxies act selfishly and blindly to maximize their utility. The outcome, however, does not meet their expectation. The utility may be worse than the achievable individual optimal, in which the proxies cooperatively decide how much cache to supply. Each proxy seems to optimize their individual benefit, but actually the system-wide behavior does not reflect the optimization of any objective. Without the whole view of the system, it is difficult to determine whether the outcome (or the Nash equilibrium) is desirable or not. According to different kinds of player in the game, either one of the following system-wide objectives can be achieved:

(1) Maximize the net profit of the NSP;
(2) Maximize the social utility among all proxies involved in the system.

To achieve the above objectives, we suggest two cooperative resource allocation games, namely *Profit Maximizing Game* and *Utility Maximizing Game.*

3.2 Profit Maximizing Game

Being the NSP, the objective is obvious: it aims to maximize its net profit in using COPACC architecture. The net profit, $E(q)$, of the NSP is defined as the revenue earned minus the reward paid to the proxies, i.e.

$$E(q) = R(q) - qP(q). \tag{9}$$

It is clear that the net profit is always zero in the total-rewarded pricing. Therefore, it is better to use other reward pricing if the NSP wants to earn some profit. As shown in Equation (9), the net profit is determined by the total cache space q supplied to the system , and the NSP can only influence the value of q by setting the price function $P(q)$ probably at the beginning of the game. Once the price function is set and publicized, the NSP has no control about the value of q, which is a Nash equilibrium converged from the moves of the proxies over many iterations.

The non-cooperative game does not lead to a unique Nash equilibrium. The main reason is that the aggregated cache space currently supplied to the network does affect the price, and thus interferes the proxy's decision. For the same price function used, the system may converge to different equilibrium. There is no guarantee for the NSP to set a particular marginal-rewarded pricing function that leads to a desirable outcome, which maximizes its net profit. To ensure the existence of a unique, predictable Nash equilibrium, we simplify the price function to a constant p, which remains the same regardless of how much cache is supplied to the network. By setting a constant price, we show that the game admits a unique Nash equilibrium, and the NSP can choose a proper price p to maximize its net profit.

With this assumption, we have a Stackelberg game[Basar and Olsder 1999] that has one leader (the NSP) and H non-cooperative Nash followers (the proxies). The NSP strategically decides the price p, and the proxies react with the best amount of cache s_i to supply. This defines a non-cooperative game between each independent proxy in the network, with the underlying solution being the Nash equilibrium. Each proxy selfishly selects s_i to satisfy its objective function:

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = s_i P(q) - C_i(s_i) = s_i p - C_i(s_i). \tag{10}$$

We assume that if the net utility of a proxy is less than or equal to zero, it will not participate in the system, and it will be removed from the list of proxies. Note that there is a boundary constraint for the variable s_i, i.e. $0 \le s_i \le \hat{S}_i$. The problem is formulated as a constrained optimization, which can be solved by the method of Lagrangian Multiplier.

Let $\{s_i{}^*\}_{i=1}^{H}$ be a set containing the amount of cache supplied by the proxies to the system such that it satisfies

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = U_i(s_i{}^*). \tag{11}$$

One can analytically find the value of s_i^* based on the value of p, using the first-order condition.

$$U_i'(s_i) = p - C_i'(s_i) = 0 \tag{12}$$

$$C_i'(s_i) = A_i \theta_i e^{\theta_i (s_i - b_i)} = p \tag{13}$$

$$s_i = \frac{ln(p/A_i\theta_i)}{\theta_i} + b_i. \tag{14}$$

By solving s_i in Equation (13), one can obtain the solution of s_i^*.

$$s_i^* = \begin{cases} 0, & s_i \le 0 \\ s_i, & 0 < s_i \le \hat{S}_i \\ \hat{S}_i, & s_i > \hat{S}_i. \end{cases} \tag{15}$$

Obviously, there is only one value of s_i^* that can satisfy the objective function in Equation (11). Thus, the game admits one and only one Nash equilibrium, i.e. there exists a unique $\{s_i{}^*\}_{i=1}^{H}$, for each value of p.

THEOREM 3.2.1. *The profit maximizing game admits one and only one Nash equilibrium*

PROOF. Consider the second-order differential equation of the utility $U_i(s_i)$,

$$U_i''(s_i) = -C_i''(s_i). \tag{16}$$

Since the cost function is defined as a strictly increasing function with convex shape, the second-order differential equation should always be positive, i.e. $C_i''(s_i) > 0$. It shows that $U_i''(s_i)$ is always less than zero, and the utility function admits at most one maximum. Thus, the strategy of proxy i is either s_i^* that satisfies the Equation (13) if the maximum located in the range of $(0, \hat{S}_i)$, or the boundary value 0 or $\hat{S}_i$. As each proxy has it own unique optimal strategy s_i^* independent of others, a unique $\{s_i{}^*\}_{i=1}^{H}$ does exist. □

Thus, given the value of price p, the NSP can predict the total cache space q^* contributed to the system, that is

$$q^* = \sum_{i=1}^{H} s_i^*. \tag{17}$$

If the NSP knows the parameters A_i, θ_i and b_i of all the proxies, it can formulate its own maximization, which aims at maximizing the net profit with respect to q^*.

$$\max_{p \ge 0} E(q^*) = R(q^*) - q^* p \tag{18}$$

Since the total amount of cache space q^* is solely depended on the value of price p through Equation (15) and (17), one can rewrite the objective function by substituting q^* in terms of p. In the COPACC system, if all the s_i do not violate the feasible constraints, the objective function can be rewritten as Equation (19).

$$\max_{p\geq 0} E(p) = R(\sum_{i=1}^{H}(\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i)) - p \cdot \sum_{i=1}^{H}(\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i) \tag{19}$$

The derivative of q^* and $E(p)$ with respect to p are shown below. The optimal price, p^*, can be obtained by solving the first-order condition in Equation (23).

$$\frac{dq^*}{dp} = \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} \tag{20}$$

$$E'(p) = A'e^{-\theta'(q^*-b')} \cdot \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} - q^* - p \cdot \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} \tag{21}$$

$$= \frac{A'}{p}(\sum_{i=1}^{H}\frac{1}{\theta_i})e^{-\theta'(q^*-b')} - q^* - \sum_{i=1}^{H}\frac{1}{\theta_i} \tag{22}$$

$$E'(p) = 0. \tag{23}$$

Although it is hard to find the close-form solution of p^* for Equation (23), one can solve this optimization efficiently using numerical method. Once the NSP find the optimal price, it can calculate the value of all s_i^* using Equation (15). If all s_i^* are inactive, i.e. they satisfy the condition $0 \leq s_i^* \leq \hat{S}_i$, the net profit of the NSP is guaranteed to be maximum by setting the optimal price to p^*.

What if some of the constraints are active, they do not satisfy the boundary condition of s_i? The problem becomes more complicated, but one can still find the optimal value of p mathematically. The solution is based on the techniques of Lagrangian multiplier. It can be shown that the objective function $E(p)$, without considering the cache constraints, is a concave function, and all the constraints regarding s_i are linearly. Thus, it is a concave programming problem, and there exists a unique solution that satisfies the KKT-condition in Equation (25)-(30).

$$L = E(p) - \sum_{i=1}^{H}\mu_i^l s_i + \sum_{i=1}^{H}\mu_i^u(s_i - \hat{S}_i) \tag{24}$$

$$\frac{\partial L}{\partial p} = \frac{\partial E(p)}{\partial p} - \sum_{i=1}^{H}\mu_i^l\frac{\partial s_i}{\partial p} + \sum_{i=1}^{H}\mu_i^u(\frac{\partial s_i}{\partial p} - \hat{S}_i) = 0 \tag{25}$$

$$\mu_i^l \geq 0, \qquad i = 1, ..., H \tag{26}$$

$$\mu_i^u \geq 0, \qquad i = 1, ..., H \tag{27}$$

$$\mu_i^l s_i = 0, \qquad i = 1, ..., H \tag{28}$$

$$\mu_i^u(s_i - \hat{S}_i) = 0, \qquad i = 1, ..., H \tag{29}$$

$$0 \leq s_i \leq \hat{S}_i, \qquad i = 1, ..., H. \tag{30}$$

We now present an algorithmic approach to find the optimal value of p, which is derived directly from the KKT-condition. Fig. 2 shows the profit maximizing algorithm for the NSP in the profit maximizing game. It first assumes that the boundary constraints of all s_i are inactive, i.e. all the cache space s_i lie between 0 and $\hat{S}_i$. Thus, the Equation (28) and (29) hold only if the μ_i^l and μ_i^u are zero. The optimization problem is now similar to the unconstrained problem in Equation (19), and we can apply the numerical method stated previously to calculate the optimal price p^* as well as all s_i. If the s_i are feasible, we have obtained the best solution. Otherwise, we know that some of the boundary constraints are violated, and the corresponding values of μ_i^l or μ_i^u are not equal to zero. In that case, the value of s_i is forced to be the boundary value (either 0 or $\hat{S}_i$) followed by Equation (28) or (29). We can identify the active constraints of s_i from the result obtained in Equation (23). If the optimal s_i found in the unconstrained optimization is less than zero, the proxy should not participate in the system. Therefore, we remove the proxy from the system by setting $s_i = 0$. If the optimal s_i is greater than the maximum capacity the proxy can provide, the proxy supplies $\hat{S}_i$ units only, and $s_i = \hat{S}_i$. After hard-setting the value of certain s_i, we execute the algorithm again to find the numerical solution for the optimal value of p. If the outcome of all s_i are feasible, we get the best solution. Otherwise, we repeat the previous steps to adjust the value of s_i and execute the algorithm until the resulted s_i are feasible. In practice, integral value of cache quantity is desired. Thus, an additional checking on $\lceil s_i \rceil$ and $\lfloor s_i \rfloor$ as the solution should be made to assure optimality.

Until now, we assume the NSP knows the characteristic of the cost function of each proxy such that it can determine the behavior of the proxies, and it can construct its own objective function. But one interesting question to ask is whether the NSP can maximize its net profit without knowing the individual cost function of each proxy. As such, the NSP can only observe the action of each proxy by setting a probing price. The NSP keeps adjusting the price gradually until a desirable profit is obtained. It is analogous to a commodity market, where the optimal price is determined through numerous iterations of refinement.

Although the NSP does not know the exact amount of total cache space q supplied to the system for each p, it is always true that increasing the price leads to a non-decreasing movement of the total cache space. Fig. 3 plots a sample relationship between the total cache space q and the price p. Both p and q move non-linearly in the same direction. Due to the boundary constraints of s_i, the function relating p and q is continuous but not differentiable. Fig. 4 plots a sample relationship between the net profit $E(p)$ and the price. We observe that the value of $E(p)$ in Equation (19) generally increase for small value of p. Then it reaches the global maximum, and decreases with increased value of p.

We now construct a *Price Establishing Protocol* for the NSP to determine the optimal price used in the system. Let's assume the proxies choose the best s_i to maximize their net utility, stated in Equation (10), based on the price p given from the NSP. The NSP keeps announcing different value of price p, and the proxies reply to the NSP with the amount of cache space it agrees to contribute. Based on the total cache space q supplied by the proxies, the NSP decides the best pricing

Profit Maximizing Algorithm:

```
declare P = { 1, 2, ... , $H$};    // indexes of proxy that it's $s_i$ has not been determined yet
declare $P^l = \{\}$;    // indexes of proxy that it's $s_i$ is zero
declare $P^u = \{\}$;    // indexes of proxy that it's $s_i$ is $\hat{S}_i$
for $i := 1$ to $H$
    $s_i = 0$;
end for
while (true) do
    $q = \sum_{i \in P} s_i + \sum_{i \in P^u} \hat{S}_i = \sum_{i \in P} [\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i] + \sum_{i \in P^u} \hat{S}_i$;
    Solve the optimal price $p$ that maximize $E(p) = R(q) - p \cdot q$ (or find $p$ s.t. $E'(p) = 0$);
    for $i := 1$ to $H$
        $s_i = ln(p/A_i\theta_i)/\theta_i + b_i$;
    end for
    if $0 \leq s_i \leq \hat{S}_i \; \forall i \in P$ then
        break;    // end the while loop
    end if
    declare $p^t = \{\}$;    // a temporary set
    for $i := 1$ to $H$
        if $s_i \leq 0$ then
            $P^t = P^t \cup \{i\}$;
        end if
    end for
    if $(P^t - P^l) \neq \emptyset$ then
        $P^l = P^t$;
        $P = P - P^t$;
        continue;    // next iteration of the while loop
    end if
    for each $i \in P$
        if $s_i \geq \hat{S}_i$ then
            $P^u = P^u \cup \{i\}$;
            $P = P - \{i\}$;
        end if
    end for
end while
return $p$;    // $p$ is the optimal price
```

Fig. 2. Profit Maximizing Algorithm for the NSP in the Profit Maximizing Game.

strategy to maximize its net profit. It can be thought as a search problem for an optimal value of p without a formal equation.

At the beginning of the protocol, the NSP makes an initial guess of the probing price, say $\dot{p}$. It announces the price to the proxies, and retrieves the corresponding value of q. The net profit $E(\dot{p})$ can be calculated based on the value of $\dot{p}$ and q. In each iteration, the NSP decides a new probing price based on the old price and the percentage change of the net profit. It then sets a new price and measure the change of the net profit as compared with the old one. The process goes on until the price converges to an optimal value, which achieves the maximum profit.

The search method stated above is the simplest one of the zero-order method (or maximization method without derivatives) in the literature. Some advanced direct search methods can also be applied in the *Price Establishing Protocol* to achieve global optimization with fast converging speed. We suggest to use Pattern Search

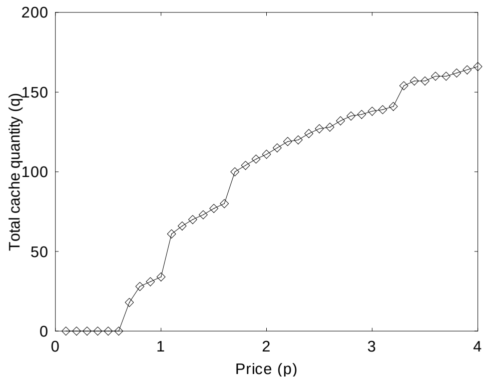


Fig. 3. A sample plot of total cache space q versus price p.

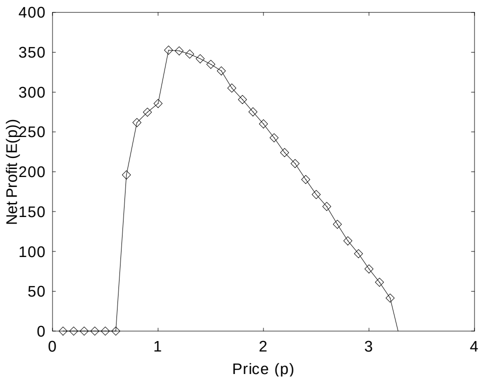


Fig. 4. A sample plot of net profit $E(p)$ versus price p.

Method [Kolda et al. 2003] to find the optimal price in the protocol. Moreover, other search method that guarantee global optimal can also be applied here.

So far we have presented the way for the NSP to maximize its net profit by setting up proper price. Sometimes, the NSP has no preference about maximizing its net profit. Instead, the proxies may prefer to maximize the social utility among themselves. In this situation, the objective of the optimization becomes maximizing the social utility, which is defined as the total net utility summed over all proxies. In what follows, we will present the utility maximizing game.

3.3 Utility Maximizing Game

Another system-wide property we would like to achieve is the social utility. It reflects the level of satisfaction of the proxies participating in the network. In this section, we present a resource allocation game that aims at maximizing the social utility of the system. Cooperation of the proxies is essential in this optimization.

The net utility $U_i(s_i)$ of each proxy supplying s_i units of cache is shown in Equation (5). We define the social utility as the individual utility summed over all proxies, which is

$$SU(s_1, s_2, ..., s_H) = \sum_{i=1}^{H} U_i(s_i). \tag{31}$$

The global objective is to maximize the social utility with respect to s_i subjected to the boundary constraints $0 \le s_i \le \hat{S}_i$, that is

$$\max_{0 \le s_i \le \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0 \le s_i \le \hat{S}_i} \sum_{i=1}^{H} [s_i P(q) - C_i(s_i)]. \tag{32}$$

As the value of q equals to $\sum_{i=1}^{H} s_i$ and the current price is calculated based on the value of q, the net utility $U_i(s_i)$ of proxy i does not solely depends on s_i supplied by itself, but also depends on the cache space s_{-i} contributed by the other proxies. The multi-variable optimization of the above objective function becomes difficult. Even the form of the partial derivative with respect to s_i is complicated. It is desirable to break down the problem into smaller subproblems, and each subproblem can be handled easier.

Fortunately, the objective function in Equation (32) can be simplified as:

$$\max_{0\leq s_i\leq \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0\leq s_i\leq \hat{S}_i} \left[q \cdot P(q) - \sum_{i=1}^{H} C_i(s_i) \right]. \tag{33}$$

The first term is equivalent to the credit rewarded to the proxies, while the second term represents the total cost of providing q units of cache by the proxies. Note that with the fixed quantity of cache space q, the first term is always constant regardless of the s_i. The social utility varies only by adjusting the allocation of s_i. Hence, the best social utility with a fixed cache quantity can be obtained when the total cost of providing the cache is minimized. The objective function in Equation (32) can be rewritten as below:

$$\max_{0\leq s_i\leq \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0\leq q\leq \hat{Q}} \left[q \cdot P(q) - \min_{0\leq s_i\leq \hat{S}_i} \sum_{i=1}^{H} C_i(s_i) \right] \tag{34}$$

$$s.t. \qquad q = \sum_{i=1}^{H} s_i \tag{35}$$

$$\hat{Q} = \sum_{i=1}^{H} \hat{S}_i. \tag{36}$$

Thus, the problem can be decomposed into two subproblems, namely *Minimal cost caching problem* and *Optimal cache quantity problem.*

(1) **Minimal Cost Caching Problem(MinCost)**: Find the minimal cost to provide q units of cache by the cooperative proxies with respect to s_i.

(2) **Optimal Cache Quantity Problem(OptQ)**: Find the quantity of cache space q that guarantee best social utility.

These two subproblems are linked together by the common variable q. In the first subproblem, given the value of q, we claim that it is always possible to find a unique set of allocation s_i that minimizes the total cost. Thus, there is a one-to-one mapping between q and $\{s_i\}_{i=1}^{H}$. Once the first subproblem is solved, the whole problem depends only on the variable q, while not the actual allocation $\{s_i\}_{i=1}^{H}$. We then solve the second subproblem by finding the optimal value of q, as well as the price $P(q)$, to generate maximum social utility.

We now present the concrete formulation and the proposed solution to the subproblems.

3.3.1 Minimal Cost Caching Problem.
As the name minimal cost caching implied, this subproblem is about finding the cheapest way to supply q units of cache space among the proxies. We are also interested in the cache space $\{s_i\}_{i=1}^{H}$ allocated by the proxies that minimize the total cost.

The minimal cost caching problem is formulated mathematically as:

$$\textbf{MinCost:}\quad MinCost(q) = \min \sum_{i=1}^{H} C_i(s_i) \tag{37}$$

$$s.t.\quad 0 \le s_i \le \hat{S}_i, \qquad i = 1, ..., H \tag{38}$$

$$\sum_{i=1}^{H} s_i = q. \tag{39}$$

The formulation like Equation (37) is rather common in the field of optimization. It can be solved algorithmically using the well-known dynamic programming method. Let $B(i, j)$ be a two-dimensional matrix that stores the minimum cost of contributing j units of cache by the first i proxies, where $1 \le i \le H$ and $0 \le j \le q$.

$$B(i,j) = \begin{cases} C_1(j), & 0 \le j \le \hat{S}_1 \\ \infty, & i = 1,\ \hat{S}_1 < j \le q \\ \min\limits_{0 \le k \le j, k \le \hat{S}_i} B(i-1, j-k) + C_i(k), & 2 \le i \le H,\ 0 \le j \le q. \end{cases} \tag{40}$$

The matrix can be filled in plane-order starting from $B(1, 0)$ to $B(H, q)$, and the latter gives the minimum cost of providing q units of cache. The corresponding s_i of each proxy can be obtained through backtracking the iterations. This dynamic programming algorithm has time complexity $O(q \cdot H \cdot M)$, where $M = \max_{1 \le i \le H} \hat{S}_i$.

Although the dynamic programming method solved the MinCost problem, it is not always desirable because it requires a powerful and dedicated node in the network to execute the algorithm centrally. As a consequence, a distributive algorithm is generally preferred to solve the problem in the network.

Before proceeding, it is a good idea to understand the mathematical solution of this cost minimization problem using optimization theory. Consider the problem stated in Equation (37), it is proven as a constrained convex optimization problem.

THEOREM 3.3.1. *MinCost is a constrained convex optimization problem.*

PROOF. To show this, we have to prove the truth of the following two statements.

(1) The feasible region of the solution space s_i under the constraints is convex.

(2) The summation of the individual cost function is a convex function.

Statement (1) follows directly from the fact that all the constraints for s_i are linear. It is obvious that the equality constraint involving q and the boundary constraints for each s_i are linear equation of s_i. Thus, the feasible region is a convex set.

Consider the Hessian matrix of the total cost function,

$$\nabla^2[\sum_{i=1}^{H} C_i(s_i)] = \begin{bmatrix} C_1''(s_1) & 0 & 0 & 0 \\ 0 & C_2''(s_2) & 0 & 0 \\ 0 & 0 & C_3''(s_3) & 0 \\ 0 & 0 & 0 & ... \end{bmatrix} \tag{41}$$

or

$$\nabla^2[\sum_{i=1}^{H} C_i(s_i)]_{ij} = \begin{cases} 0, & i \neq j \\ C_i''(s_i), & i = j. \end{cases} \tag{42}$$

Since the cost function $C_i(s_i)$ is strictly increasing, the value $C_i''(s_i)$ are always positive. The Hessian matrix is positive semi-definite. Thus, the total cost function is a convex function. Statement (2) holds. □

Convexity is a nice property in constrained optimization. In a convex optimization, the local minimum is indeed the global minimum. As a result, the KKT-condition is the sufficient and necessary condition for optimality. In other words, a set of $\{s_i\}_{i=0}^{H}$ that satisfies the KKT-condition is the solution to our cost minimization problem. The KKT-condition for the MinCost problem is shown below:

$$L(s_i) = \sum_{i=1}^{H} C_i(s_i) - \lambda(\sum_{i=1}^{H} s_i - q) - \sum_{i=1}^{H} \mu_i^l s_i + \sum_{i=1}^{H} \mu_i^u (s_i - \hat{S}_i) \tag{43}$$

$$\frac{\partial L}{\partial s_i} = C_i'(s_i) - \lambda - \mu_i^l + \mu_i^u = 0, \qquad i = 1, ..., H \tag{44}$$

$$\mu_i^l \geq 0, \qquad i = 1, ..., H \tag{45}$$

$$\mu_i^u \geq 0, \qquad i = 1, ..., H \tag{46}$$

$$\mu_i^l s_i = 0, \qquad i = 1, ..., H \tag{47}$$

$$\mu_i^u (s_i - \hat{S}_i) = 0, \qquad i = 1, ..., H \tag{48}$$

$$0 \leq s_i \leq \hat{S}_i, \qquad i = 1, ..., H \tag{49}$$

$$\sum_{i=1}^{H} s_i = q. \tag{50}$$

The MinCost is a convex optimization probem, meaning that there exists a unique solution $\{s_i\}_{i=1}^{H}$ satisfying the Equation (44)-(50). The optimal cache allocation s_i can be determined by solving the set of linear equations. We are now ready to present our distributed approach to the cost minimization problem. The algorithm is emerged from the mathematics above.

We start with a simplified version of the MinCost problem in Equation (37), having the storage constraints removed from each proxy. It becomes an equality constrained problem, which can be solved by the method of Lagrange multiplier. The necessary condition is similar to the KKT-condition stated previously, but with the equations involving μ_i^l and μ_i^u omitted. Note that the plus or minus sign of the multiplier term does not affect the solution.

$$L(s_i) = \sum_{i=1}^{H} C_i(s_i) - \lambda(\sum_{i=1}^{H} s_i - q) \tag{51}$$

$$\frac{\partial L}{\partial s_i} = C_i'(s_i) - \lambda = 0, \qquad i = 1, ..., H \tag{52}$$

$$\sum_{i=1}^{H} s_i = q. \tag{53}$$

We instantiate the cost function according to the COPACC system. Given Equation (52), we can derive s_i in terms of λ.

$$C_i'(s_i) = A_i \theta_i e^{\theta_i (s_i - b_i)} = \lambda \tag{54}$$

$$s_i = \frac{ln(\lambda / A_i \theta_i)}{\theta_i} + b_i. \tag{55}$$

The variable λ is called shadow price, which is introduced by the proxies to establish implicitly the best cache allocation among them. The equation of s_i is similar to the one shown in Equation (14), but in here we have one more condition about the total cache quantity (in Equation (53)) to hold. By substituting s_i to Equation (53), we can solve the value of λ, and thus, the values of all s_i.

$$\sum_{i=1}^{H} \left[\frac{ln(\lambda / A_i \theta_i)}{\theta_i} + b_i \right] = q \tag{56}$$

$$\lambda = e^{\frac{q + \sum_{i=1}^{H} [ln(A_i \theta_i)/\theta_i - b_i]}{\sum_{i=1}^{H} 1/\theta_i}}. \tag{57}$$

If we have all the parameters about the individual cost function of each proxy, we can find the optimal λ as well as the s_i directly. However, in the distributed approach, we must rely on iteratively refining the value of λ until the optimal value is reached. The resulted total cache space is used as an indicator for optimality. The minimal cost is achieved when $\sum_{i=1}^{H} s_i$ is equal to q. In order to use the distributive algorithm, we assume the proxies are cooperative, and they do follow the cost minimization protocol to determine the amount of cache contributed to the system. The protocol runs collaboratively with assistance from a proxy coordinator.

The coordinator first make an initial guess of the shadow price λ, and notifies the proxies. Each proxy reacts to the shadow price with a s_i obtained by Equation (55), which is privately known to the proxy. Then, the coordinator updates the shadow price based on the total cache space contributed to the system. If the total cache space is more than required, i.e. $\sum_{i=1}^{H} s_i > q$, the shadow price is set too high, and it should be reduced. If the cache supply is insufficient, the shadow price should be increased. The process continues until the optimal value is achieved, where the total cache supplied matches the requirement, i.e. $\sum_{i=1}^{H} s_i = q$.

The most crucial part remained is how to update the shadow price according to the cache supplied. The updating rule should be selected carefully as it determines the effectiveness of the protocol. Since the problem is proven to exist only one minimum, even the simplest numerical search method guarantees optimal solution. Other advanced search method, of course, can be used to obtain the same result.

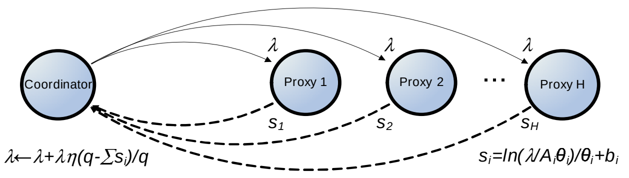


Fig. 5. Mechanism of the cost minimization protocol.

Fig. 5 shows the mechanism of the cost minimization protocol. We adopt to a simple updating rule, which increase/decrease the value of λ in proportional to the difference between the desirable cache space q and the total contributed cache from the proxies. The updating rule of λ is

$$\lambda \leftarrow \lambda + \lambda\eta(\frac{q - \sum_{i=1}^{H} s_i}{q}). \tag{58}$$

The learning rate, η, is a factor that controls the converging speed and the accuracy of the protocol. A large value of η is used initially to speed up the convergence, and it starts to decrease gradually in order to obtain an accurate solution. The protocol is executed periodically to ensure that the cost remains minimal after any join or leave of the proxies. In steady state, the λ leads to a cache allocation with minimal cost.

The solution of this simplified MinCost problem can be extended to the original problem with the cache constraints. The participating proxies react to the shadow price similarly as in the simplified MinCost problem. The only difference is that the proxy coordinator has an additional task to determine whether the cache constraints of the proxies are violated.

Consider the Equation (47) in the KKT-condition, either μ_i^l equals zero or s_i equals zero. Similarly in Equation (48), either μ_i^u equals zero or s_i equals $\hat{S}_i$. To solve the set of linear equations, we have to examine whether μ_i^l and μ_i^u are equal to zero. Assume both μ_i^l and μ_i^u are zero, the formulation of the original problem reduces to the simplified version. We apply the cost minimization protocol to obtain the best cache allocation for a total of q units of cache space. However, the resulted s_i may not satisfy the boundary constraints specified in Equation (49). In that case, depending on the value of s_i, one of the μ_i^l and μ_i^u is not zero. If s_i is less than or equal to zero for certain proxy i, we are sure that this s_i has optimal value of zero, and the corresponding μ_i^l is not zero. The proxy is not eligible for contributing as the cost of supplying the cache is comparatively high. Similarly, if s_i is greater than $\hat{S}_i$, we are sure that the s_i of the proxy has optimal value of $\hat{S}_i$. This proxy should provide as much cache as possible since the cost is comparatively low. Thus, we can eliminate some proxies, whose value of s_i is known already, from the problem formulation and resolve the s_i for the remaining proxies. Note that the total required cache space q of the eliminated problem should be updated accordingly, by subtracting the $\hat{S}_i$ of the oversupplied proxies. The algorithm for the cost minimization protocol used by the coordinator is shown in Fig. 6.

By the cost minimization protocol with a given fixed total cache quantity, the proxies can cooperatively allocate the best amount of cache space to achieve minimal

Cost Minimization:

```
declare P = { 1, 2, ... , $H$};    // indexes of proxy that it's $s_i$ has not been determined yet
declare $P^l$ = {};    // indexes of proxy that it's $s_i$ is zero
declare $P^u$ = {};    // indexes of proxy that it's $s_i$ is $\hat{S}_i$
while (true) do
  the new cache requirement $q' = q - \sum_{i \in P^u} \hat{S}_i$;
  solve the Simplified MinCost Problem distributively with the cache requirement of $q'$
   among proxy $i \in P$ and get the optimal $s_i$ for proxy $i \in P$;
  if $\forall i \in P$, $0 \le s_i \le \hat{S}_i$ then
    break;    // end the while loop
  end if
  declare $p^t$ = {};    // a temporary set
  for each $i \in P$
    if $s_i \le 0$ then
      $P^t = P^t \cup \{i\}$;
    end if
  end for
  if $P^t \neq \emptyset$ then
    $P^l = P^l \cup P^t$;
    $P = P - P^t$;
    continue;    // next iteration of the while loop
  end if
  for each $i \in P$
    if $s_i \ge \hat{S}_i$ then
      $P^u = P^u \cup \{i\}$;
      $P = P - \{i\}$;
    end if
  end for
end while
```

Fig. 6. Cost Maximization Protocol for the proxy coordinator.

cost.

3.3.2 **Optimal Cache Quantity Problem**. The next problem is to determine the optimal amount of cache quantity. The optimal cache quantity problem refers to the problem of finding the total quantity that results in maximum social utility. Let $M(q)$ be the minimum cost of providing q units of total cache. For each q, the value of $M(q)$ can be evaluated by solving the corresponding MinCost problem, using the cost minimization protocol. The OptQ problem can be formulated as

$$\textbf{OptQ:} \quad \max_{0 \le q \le \hat{Q}} [q \cdot P(q) - M(q)] \tag{59}$$

$$where \ \hat{Q} = \sum_{i=1}^{H} \hat{S}_i.$$

Obviously, the objective function depends on the variable q only, where $P(q)$ is a decreasing function of q and $M(q)$ is an increasing function of q (see Fig. 7). In fact, the objective function may contain multiple maxima, depending on the cost functions and revenue function used in the system. Fig. 8 plots the revenue, minimum cost and social utility with respect to cache quantity in the COPACC

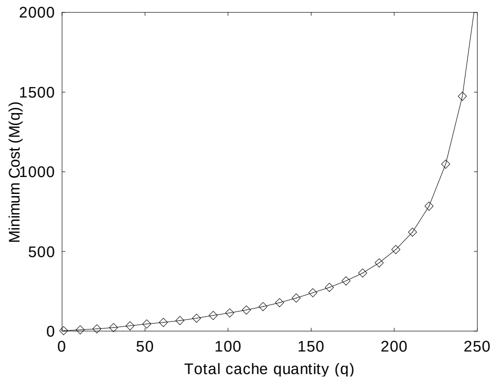


Fig. 7. A sample plot of minimum cost versus cache quantity q.

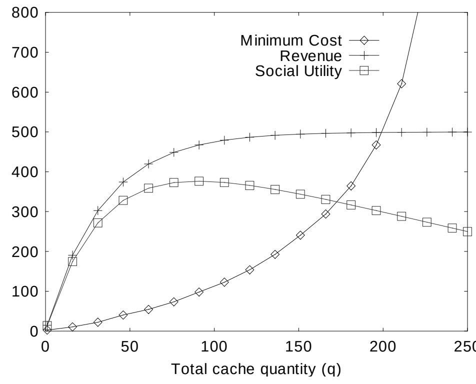


Fig. 8. A sample plot of social utility versus cache quantity q.

system. In this example, the social utility is calculated using the total-rewarded price, and the maximum is achieved when the cache quantity is around 75.

Since we do not have the close form solution for the MinCost problem, we cannot rely on any optimization method that involves derivative of the objective function. In order to find the optimal cache quantity, we suggest to use direct search method, which is similar to the one used in the profit maximizing game. Pattern search with multiple initial guesses is a good approach to the optimal cache quantity problem.

In the pattern search method, an initial step size s is chosen and the search is initiated from a starting point q. The method involves the steps of exploration and pattern search. In the exploration step, it tries to probe the value of the social utility by increasing or decreasing the cache quantity. Let $q' = q$, the objective function is evaluated at $q' + s$. If the value increases, then q' is updated to $q' + s$. Otherwise, the function is evaluated at $q' - s$. If the value increases, q' is updated to $q' - s$. In case both of them fail in the test, the original value of q' is retained. An exploration is said to be successful if the function valued at q' is higher than q by a predetermined amount. The pattern search algorithm starts from a quantity q. The exploration step is made in q. If the exploration fail, the step size is reduced by a factor of r, i.e. $s \leftarrow rs$. Otherwise, a new base point of q is established according to the exploration. The search continues until the cache quantity q converged. To avoid trapping in the local maxima at the steady state, the proxy coordinator should periodically probe the system to see if the cache quantity is still optimal.

The solution of the optimal cache quantity problem is indeed the optimal quantity for the original utility maximizing problem. It guarantees maximal social utility in the network. Moreover, the optimal cache space supplied by each individual proxy is determined through the cost minimization protocol, and the price is set according to the price function.

In this section, we have presented the utility maximizing game, which aims at maximizing the social welfare of the proxies in the network. The performance of the three resource allocation games are being evaluated in the next section.

Proxy i	A_i	θ_i	b_i	$\hat{S}_i$	Cost
1	10	0.06	0	50	Normal
2	10	0.06	15	50	Low
3	15	0.08	0	50	High
4	8	0.12	10	50	Low for small quantity, high for large quantity
5	10	0.04	0	50	High for small quantity, low for large quantity

A'	θ'	b'
15	0.03	0

Table II. Parameters used in the resource allocation game.

4. PERFORMANCE EVALUATION

The main focus of this section is to evaluate the effectiveness of the proposed revenue-rewarding scheme in encouraging the participation of the NSP and the proxies. We show that the scheme can provide a strong incentive for different entities to join the system. We have examined the use of revenue-rewarding in the three resource allocation games, i.e. non-cooperative game (*NonCoop*), profit maximizing game (*ProfitMax*), and utility maximizing game (*UtilMax*). We have studied the net profit of the NSP as well as the social utility of the proxies in the games. We have also compared the individual utility of the proxies in each game. The results demonstrate that under different resource allocation games, different level of incentive are given to different entities in the network. Moreover, an economical cache supply is achieved in the ProfitMax and the UtilMax, where the "good" peers, which have cheaper cost in providing cache, are retained to participate in the system.

Unless otherwise specified, the following default settings were used in the evaluation. We considered a proxy caching network consisting of five proxies, which operated under the same NSP. The revenue function was approximated by the experimental results from the cost reduction in the COPACC system. Each proxy was assigned with an exponential cost function. The parameters used for the cost and revenue functions are shown in Table II, and the corresponding functions are plotted in Fig. 9. Note that proxy 1,2 and 3 have similar cost function with different level of expensiveness. Proxy 4 supplies cache with low initial cost but high variable cost. In contrast, proxy 5 has high fixed cost but low variable cost. For simplicity, each proxy has the same storage capacity. Lastly, the marginal-rewarded pricing was applied in the non-cooperative game and the utility maximizing game.

The evaluation of different rewarding schemes were done based on mathematical simulation, which was implemented in MATLAB 7.0. The built-in Pattern Search Tool, provided in the Genetic Algorithm and Direct Search Toolbox of MATLAB, was used to solve the search problem.

4.1 Convergence

The first issue we are looking at is the convergence. It is a basic requirement that the resource allocated by the proxies should converge in each game. We analyze the behavior of each game based on the cache contributed at the steady state. Fig. 10 depicts the quantity of cache supplied by the five proxies in each iteration. It demonstrates that all three resource allocation games converge to a steady state

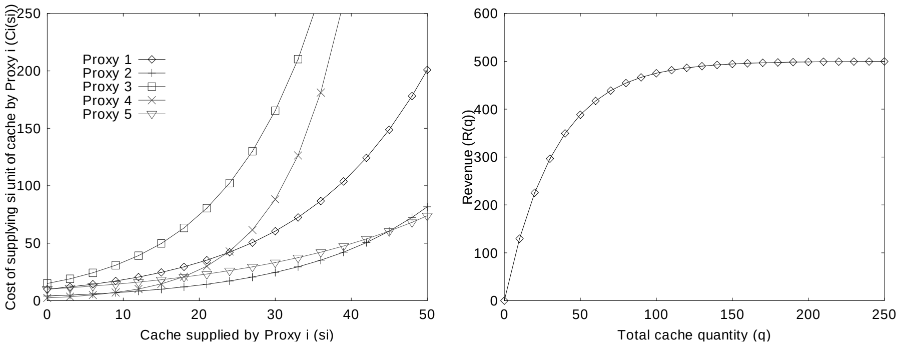


Fig. 9. Cost and revenue functions being used in the evaluation.

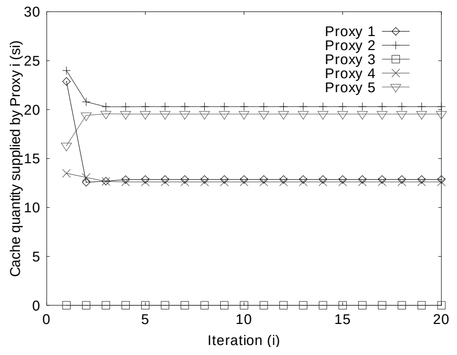
after a number of iterations. It is observed that the NonCoop converges fast, while it takes more iterations for the ProfitMax and the UtilMax to stabilize. For the ProfitMax and the UtilMax, the speed of convergence is depended on the direct search method implemented. In the pattern search method, a large step size is used initially, therefore, the cache quantity varies dramatically at the beginning. As the step size decreases gradually, the cache quantity stabilizes and converges to the optimal value. Fortunately, the converging speed does not affect the performance of the game. As long as the search method converges to an optimal value, the corresponding objective function is optimized, and the goal is achieved.

4.2 Participation Incentive

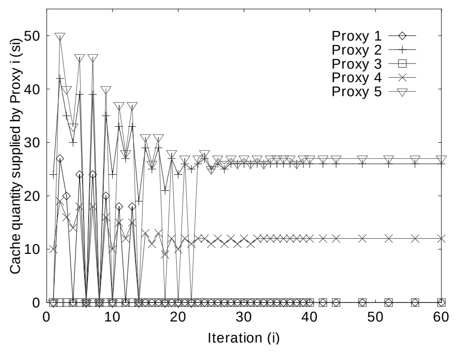
The primary design objective of this work is to present an incentive mechanism to encourage participation of the entities in the network. The evaluation results show that this incentive mechanism applied in the COPACC system provides a strong incentive for both the NSP and the proxies. In addition, the incentive for the NSP is different from that of the proxies. The NSP is motivated by the attractive net profit to setup the COPACC system in its network, while the proxies are encouraged by the positive net utility to supply cache to the system. Hence, in this subsection, we evaluate the two incentives in the three resource allocation games.

4.2.1 **Net Profit**. Fig. 12 plots the net profit of the NSP in the three resource allocation games. As we expected, the profit maximizing game generated the highest net profit among the three games. The net profit of the ProfitMax was 352.8, which was 21% higher than that of the NonCoop, and it was 2.26 times of the net profit in the UtilMax. The UtilMax performed the worse because it tried to maximize the benefit in other dimension, social utility, by trading off the net profit.

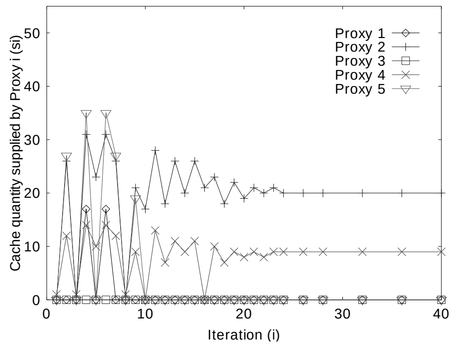
We also evaluated the performance of the three games under systems having different revenue function. All the revenue functions had the same ratio of A'/θ', but the value of θ' varied from 0.01 to 0.08. Remember that the larger the θ', the higher the revenue is for the same quantity of cache. The net profit of the NSP in the systems with different revenue function is plotted in Fig. 13. The result further illustrates that the ProfitMax achieves the highest net profit among the three games.



(a) Non-cooperative Game

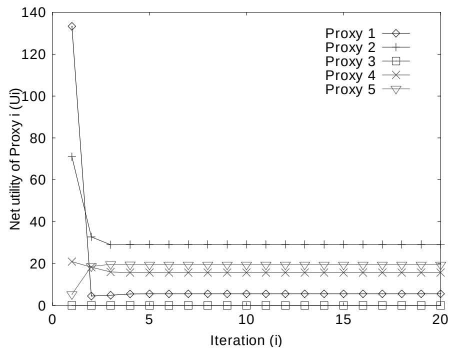


(b) Profit Maximizing Game

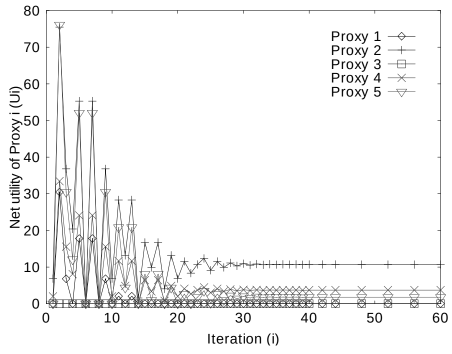


(c) Utility Maximizing Game

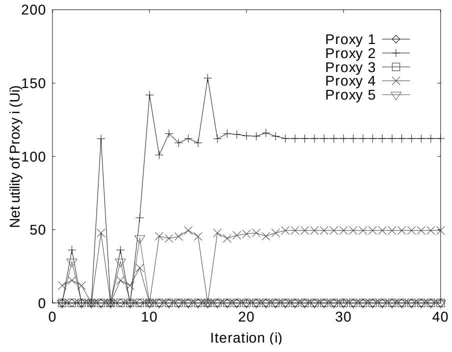
Fig. 10. Quantity of cache supplied by each proxy.



(a) Non-cooperative Game



(b) Profit Maximizing Game



(c) Utility Maximizing Game

Fig. 11. Net utility of each proxy.

Note that the revenue earned by the NSP in each game was different, depending on the cache quantity supplied to the system. The revenue, in terms of cost reduction, obtained in the COPACC system under the ProfitMax was 426.7. It is attractive to the NSP that by rewarding 17% (i.e. 73.60) of the total revenue to the proxies, which were willing to participate due to positive net utility, the NSP can enjoy 83% of the revenue as its net profit. Hence, we conclude that the profit maximizing game provides a strong incentive for the NSP to setup the revenue-

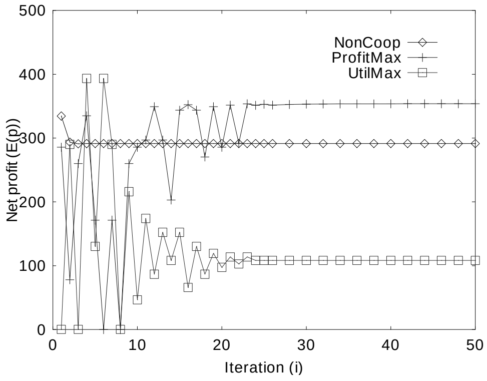


Fig. 12. Net profit in each resource allocation game.

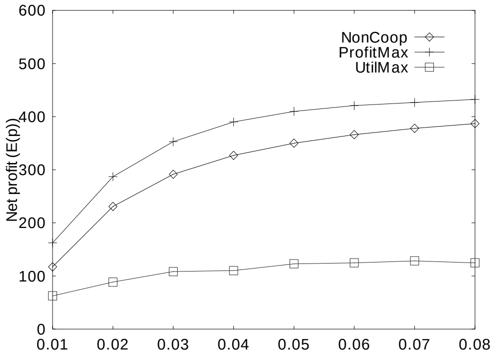


Fig. 13. Net profit of the NSP in the system with θ' varied from 0.01 to 0.08.

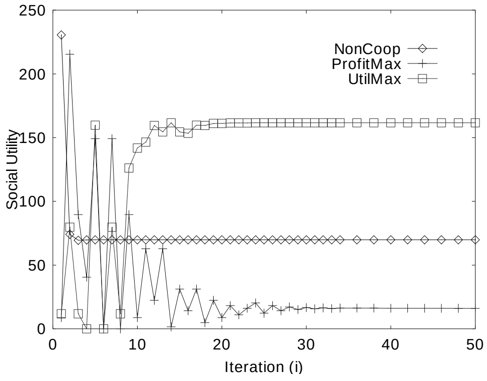


Fig. 14. Social utility in each resource allocation game.

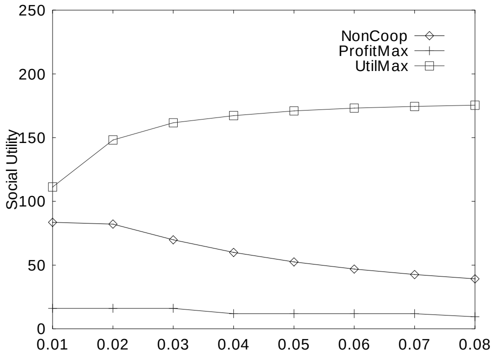


Fig. 15. Social utility in the system with θ' varied from 0.01 to 0.08.

rewarding scheme in the COPACC system. We argue that even for the UtilMax, the NSP still has incentive to deploy the incentive mechanism as it can retain 37% (i.e. 108.28) of the revenue as its profit.

4.2.2 **Social Utility**. In this subsection, we evaluate the social utility in different games. Fig. 11 demonstrates the individual utility of the proxies under different games. Only the proxies having positive net utility supplied cache to the system. It illustrates that the positive net utility provides an initiative incentive to the rational proxies to cooperate.

Fig. 14 shows the social utility of all proxies in the three games. The social utility achieved by the UtilMax was the highest among the three. In this example, the social utility of the UtilMax was 161.7, which was much higher than that of the ProfitMax (16.1) and the NonCoop (69.9). In Fig. 15, the social utility of different games were examined under the systems with different revenue function. The result also agrees that the UtilMax outperforms other games in utility maximization.

Since the ProfitMax is designed to maximize the net profit by trading off the

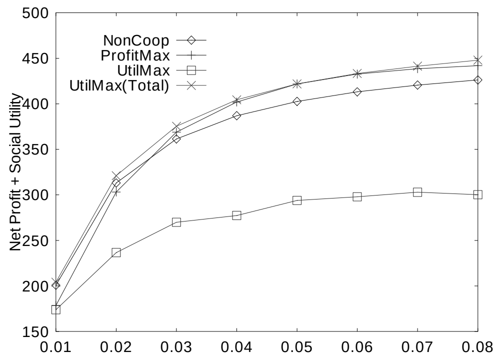


Fig. 16. Sum of the social utility and the net profit in each resource allocation game with θ' varied from 0.01 to 0.08.

social utility, the utility achieved in the ProfitMax is the lowest. Note that the social utility of the ProfitMax and the NonCoop decreased with an increase of θ'. It shows that more utility is traded for the net profit when the θ' is large.

We are also interested in the total reward received by the proxies and the cost in providing the cache. As the UtilMax performs better in utility maximizing, we focus this specific game. The total reward granted to the proxies was 182.1, and among those, 20.4 (around 11% of the reward) was used to maintain the cache. The remaining of the reward (about 89% of the total) was owned by the proxies. When we consider the return on investment, which is defined as the ratio between the utility and the cost, i.e. 7.9, it can definitely encourage the proxies to participate. In contrast, the return on investment of the ProfitMax was 0.28. Thus, under the UtilMax, the proxies have a strong incentive to contribute cache in the system.

4.2.3 **Discussion**. In fact, the ProfitMax and the UtilMax have different design objective: the former one optimizes the net profit, while the later one optimizes the social utility. The key discussion here is which approach, the ProfitMax or the UtilMax, is better. There is no strict answer to this question. It depends on the objective of implementing the incentive mechanism, and whether to benefit the NSP or the proxies. There is a trade-off between the net profit and the social utility.

Indeed, one can compare the performance of the ProfitMax and the UtilMax by looking at the sum of the social utility and the net profit as shown in Fig. 16. It can be seen that the UtilMax is not performing well as compared to the ProfitMax and the NonCoop. The reason is that under marginal-rewarded pricing, the price of resource drops quickly as the quantity increases. As a consequence, the UtilMax tries to keep a high price by avoiding large quantity of cache supplied to the system. Thus, only a small revenue is obtained, and the achievable sum of the social utility and the net profit becomes less. We also examined the UtilMax using the total-rewarded pricing, in which the net profit of the NSP is always zero. The UtilMax(Total) showed in Fig. 16 demonstrates that by using total-rewarded pricing, it achieves the best performance. In fact, we can show that the social utility obtained in UtilMax(Total) is equivalent to the maximal achievable sum of

	Cache supplied			Cost per unit cache		
	NonCoop	ProfitMax	UtilMax	NonCoop	ProfitMax	UtilMax
Proxy 1	13	0	0	1.68	-	-
Proxy 2	21	26	20	0.68	0.74	0.67
Proxy 3	0	0	0	-	-	-
Proxy 4	13	12	9	0.88	0.85	0.79
Proxy 5	20	25	0	1.11	1.09	-
Overall system	67	63	29	1.04	0.9	0.71

Table III. Cost per unit cache supplied by the proxies in different game.

the social utility and the net profit. Consider the objective function of maximizing the sum of the social utility and the net profit, that is

$$\max_{0 \le s_i \le \hat{S}_i} \left[E(q) + \sum_{i=1}^{H} U_i(s_i) \right] = \max_{0 \le s_i \le \hat{S}_i} \left[R(q) - \sum_{i=1}^{H} C_i(s_i) \right]. \qquad (60)$$

Since $P(q) = R(q)/q$ in the total-rewarded pricing, the optimization objective of the UtilMax(Total) is equivalent to the above objective. Hence, it is the maximal achievable social utility.

In general, the ProfitMax is suitable for the system that contains a centralized authority, like the NSP, and the UtilMax is good for a non-coordinated P2P-like application. By applying the revenue-rewarding scheme, the network entities are stimulated to participate in the system.

4.3 Cost effectiveness

It can be seen in Fig. 10 that not all the proxies playing in the resource allocation game participate in the system at the steady state. In fact, all the proxies in the network used the same strategy to decide the amount of cache to contribute, excepted that they had heterogenous cost function. In this subsection, we study how the cost function influences the behavior of the proxies. We show that only the cost-effective proxies contribute cache to the system.

Table III lists the quantity of cache supplied in each game. The quantity of cache admitted in each game was different. The NonCoop admitted the largest amount of cache, while the UtilMax admitted the smallest. It is due to the underlying game rule, which induces the "best" quantity (or price) of cache for the system. Clearly, the overall cost for a unit cache in the UtilMax should be the lowest because it equips with a cost minimization protocol to achieve the lowest cost. In addition, the UtilMax tends to employ few proxies, which have low cost among the proxies, to participate. Hence, we conclude that the UtilMax provides a cost-effective resource supply to the system.

We further investigate the cost of maintaining the cache for each individual proxy, which is listed in Table III. Actually, the three resource allocation games implicitly choose the best proxies to cooperate. We observed that Proxy 3 was rejected in all the games because its cost was the most expensive. Proxy 2 had a low cost function, thus, it contributed cache in all the games. Proxy 4 only contributed small amount of cache as the cost for large quantity was high. In contrast, Proxy 5 supplied large quantity due to the lowest cost. These results illustrated a desirable property of the system: the game automatically admits the best set of proxy to contribute, depending on the heterogenous cost function adopted by the proxies.

In summary, the evaluation results demonstrated that the proposed revenue-rewarding scheme applied in incentive-based COPACC system provides a strong incentive for both the NSP and the proxies to participate in the system, and the gaming approach yields a cost-effective resource allocation from the proxies.

5. RELATED WORK

Recently a lot of efforts have been made to address the problems of *free-riding* and *tragedy of the commons* [Hardin 1968] in the cooperative network. Various incentive mechanisms have been proposed to encourage the selfish nodes to cooperate by sharing their own resources with the community.

Differential service-based incentive has been well studied in the literature. Under such scheme, the peers that contribute more resource receive better quality of service, while the selfish peers contributing less are discriminated. [Buragohain et al. 2003] have suggested a game theoretic framework to improve the system's performance by eliminating non-cooperative users. In this model, the requests from a user with large contribution has a higher probability to be served. In [Ma et al. 2004], the authors have proposed a service differentiated scheduling policy that allocates bandwidth according to the peer's contribution. It showed that the social welfare is maximized when all peers have the same contribution value. [Habib and Chuang 2004] have suggested to differentiate the service in peer selection process of P2P streaming. By using the rank-based peer-selection mechanism, the contributors are rewarded with flexibility and choice in peer selection, which results in high quality streaming. For the free-riders, the options in peer selection are limited, and hence they receive low quality streaming.

Another well-known incentive model is *Reputation-based incentive*. The reputation reflects a peer's overall contribution to the network. The peers with high reputation value have extra privilege over the others. Reputation can also be used to identify how reliable and trustful a peer is. In fact, this kind of incentive has already been deployed in the KaZaA file sharing system [Kazaa], which is called the participation level. It is defined base on the megabytes the user transferred and the integrity of the files served. Downloading priority is given to the users with high reputation score. In [Gupta et al. 2003], the authors have suggested two alternative computation mechanisms to compute dynamically the reputation score of each peer in the network. The reputation score gives a general idea of the peers' level of participation in the system. The peers having high reputation is more likely to obtain better service. Based on the reputation system, [Ye and Makedon 2004] have suggested how to monitor the users behavior in a streaming network, and it tried to maintain a satisfactory level of service for the collaborative peers. [Feldman et al. 2004] have used the generalized prisoner's dilemma to model the system, and they have proposed a family of incentive techniques. A history of a peer's actions is mapped to a decision whether to cooperate with or defect on that peer. The strategies, consisting of: 1) A decision function; 2) Action history; 3) A server selection mechanism; and 4) A stranger policy, were designed to maximize both individual and social benefit. Similar approach was adopted in [Jiang et al. 2003]. They used iterated prisoner's dilemma to model the peers' interaction, and proposed a reputation-based trust model with incentive mechanism incorporated.

Our work is different from the above schemes that we follow the *Contribution reward-based incentive* approach, where monetary reward is given to the peers in proportional to their contribution. [Golle et al. 2001] have proposed a micro-payment mechanism to reward users for upload. Game theoretic model was used to analyze the equilibrium of user's strategy under several different payment schemes. The results demonstrated that the users are encouraged not only to upload files, but also to share new files to the P2P system. In [Tamilmani et al. 2004], a credit-based trading mechanism have been presented for P2P file sharing. In the model, peers ,who exchange pieces of a file, use a pairwise currency to reconcile trading differences with each other. As a result, the peers who set high upload rates receive high download rates in return. The authors also proposed a trading strategy that is good for both the network as a whole and the peers employing it. The monetary scheme provides a clean economic model for the incentive mechanism. However, it is argued to be impractical in P2P system, where a reliable accounting infrastructure has to be established to track the transactions between every peers. In contrast, it's application in our coordinated system with centralized authority is viable because the payment is made in a single direction only, i.e. from the NSP to the proxies. We are aware of a similar work in [Wang and Li 2003], which also considered revenue rewarding to the contributed peers. They model the P2P system as a Cournot Oligopoly game and used control-theoretic to maximize individual net gain. System performance requirements, like storage utilization and bandwidth stress, were considered as the global desirable properties, and they were incorporated in the dynamic payoff function of the proxies. Our work is different from it as we model the system as a Stackelberg game, and we focus on maximizing the net profit and social utility in the network.

Our work relies on pricing to regulate users' contribution. The pricing aspects of P2P network have received little attention so far. Previous research appears mainly focus on server-client model. Game-theoretic and economic model were applied to predict the influence of the price to the users' behavior. Some pricing mechanisms were suggested to maximize the revenue and the social welfare in the network. A charge-per-usage pricing model was studied in [Basar and Srikant 2002], where the users are charged for their bandwidth usage. By analyzing the strategies of the users toward the price, the optimal price is computed to maximize the revenue of the service provider. It also showed that the pricing scheme provides an incentive for the service provider to increase the network capacity. [Campos-Nanez and Patek 2003] have proposed an adaptive pricing strategy that adjusts the price in realtime manner, and the objective is again to maximize the revenue for the service provider. Their work assumed *prior* knowledge about the user arrival pattern, and thus it may not be appropriate for the P2P system with highly dynamic nodes. On the other hand, [He and Walrand 2005] have proposed a fair revenue-sharing policy, based on the weighted proportional fairness criterion, to distribute profit between cooperative provider. The fair allocation policy encourages collaboration among the providers, and hence produces higher profit for all the providers. We also adopt the proportional fairness in rewarding the revenue to the proxies. [Gupta and Somani 2004] described a pricing strategy for carrying out lookups in P2P networks. Both the resource provider and intermediate nodes, which assists in routing, are compensated so as to cover their cost of providing service. Vickrey auction, where the highest bidder wins the auction by paying the second highest bid, is used by the nodes to determine the price of the resource. The proposed protocol ensures that the rewards received by the involved nodes are maximized. We apply similar approach to reward the contributors in the cooperative network, but suggest different pricing strategy to achieve different objective.

6. CONCLUSION

In this paper, we have introduced a revenue-reward mechanism to address the incentive issue of a cooperative proxy caching system for media streaming. We research the problem of what motivates each proxy to provide cache space and how much cache space should be allocated. We have suggested a revenue-rewarding scheme to encourage proxy cooperation. In this scheme, credits are granted to the proxies for their contribution. Game theoretic model is used to analyze the interactions between proxies under different resource allocation games. It is shown that no system-wide property is achieved in a non-cooperative environment. Thus, we have further proposed two cooperative resource allocation games that lead to two different optimal situations: Maximized net profit and Maximized social welfare. Both centralized and distributed algorithms are presented for the games to achieve different optimal situation.

We have evaluated the performance of the incentive mechanism under different game settings. Our key findings can be summarized as follows:

(1) The monetary incentive scheme, revenue-rewarding, strongly motivates the network entities to cooperate in the system.

(2) The non-cooperative environment is undesirable, while the two cooperative games achieve different system-wide objectives: Net profit and Social utility.

(3) The two cooperative games yield a cost-effective resource allocation from the proxies.

REFERENCES

Adar, E. and Huberman, B. A. 2000. Free riding on gnutella. *First Monday*.

Basar, T. and Olsder, G. J. 1999. Dynamic noncooperative game theory. *SIAM Series in Classics in Applied Mathematics*.

Basar, T. and Srikant, R. 2002. Revenue-maximizing pricing and capacity expansion in a many-users regime. In *Proceedings of IEEE INFOCOM 2002*.

BitTorrent. http://www.bittorrent.com.

Buragohain, C., Agrawal, D., and Suri, S. 2003. A game theoretic framework for incentives in p2p systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P'03)*. Sweden.

Campos-Nanez, E. and Patek, S. D. 2003. On-line tuning of prices for network services. In *Proceedings of IEEE INFOCOM 2003*.

Feldman, M., Lai, K., Stoica, I., and Chuang, J. 2004. Incentive techniques for peer-to-peer networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*.

Golle, P., Leyton-Brown, K., Mironov, I., and Lillibridge, M. 2001. Incentives for sharing in peer-to-peer networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC'01)*. Tampa, Florida.

Gupta, M., Ammar, M., and Judge, P. 2003. A reputation system for peer-to-peer networks. In *Proceeding of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*.

Gupta, R. and Somani, A. K. 2004. A pricing strategy for incentivizing selfish nodes to share resources in peer-to-peer (p2p) networks. In *Proceedings of IEEE International Conference on Networks*. Singapore.

Habib, A. and Chuang, J. 2004. Incentive mechanism for peer-to-peer media streaming. In *Proceedings of International Workshop on Quality of Service (IWQoS '04)*.

Hardin, G. 1968. The tragedy of the commons. *Science 162*, 1243–1248.

He, L. and Walrand, J. 2005. Pricing and revenue sharing strategies for internet service providers. In *Proceedings of IEEE INFOCOM 2005*. Miami, USA.

Ip, A. T. S., Liu, J., and Lui, J. C. S. 2005. Copacc: A cooperative proxy-client caching system for on-demand media streaming. In *Proceedings of IFIP Networking 2005*.

Jiang, J., Bai, H., and Wang, W. 2003. Trust and cooperation in peer-to-peer systems. In *Proceedings of Grid and Cooperative Computing (GCC 2003)*.

Kazaa. http://www.kazaa.com.

Kolda, T. G., Lewis, R. M., and Torczon, V. 2003. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Review 45,* 3, 385–482.

Lui, S., Lang, K. R., and Kwok, S. 2002. Participation incentive mechanisms in peer-to-peer subscription systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*.

Ma, R. T. B., Lee, S. C. M., Lui, J. C. S., and Yau, D. K. Y. 2004. Incentive resource distribution in p2p networks. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS 2004)*. Tokyo, Japan.

Mohamed M. Hefeeda, A. H. and Bhargava, B. 2003. Cost-profit analysis of a peer-to-peer media streaming architecture. *Technical report, CERIAS TR 2002-37, Purdue University*.

Ranganathan, K., Ripeanu, M., Sarin, A., and Foster, I. 2003. To share or not to share: An analysis of incentives to contribute in collaborative file sharing environment. In *Workshop on Economics of Peer-to-Peer Systems 2003*.

Tamilmani, K., Pai, V., and Mohr, A. 2004. Swift: A system with incentives for trading. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*.

Wang, W. and Li, B. 2003. To play or to control: A game-based control-theoretic approach to peer-to-peer incentive engineering. In *Proceedings of the International Workshop on Quality of Service (IWQoS '03)*.

Ye, S. and Makedon, F. 2004. Collaboration-aware peer-to-peer media streaming. In *Proceedings of the ACM International Conference on Multimedia*. New York, USA.

Zhang, X., Liu, J., Li, B., and Yum, T.-S. P. 2005. Donet/coolstreaming: A data-driven overlay network for live media streaming. In *Proceedings of IEEE INFOCOM 2005*. Miami, USA.