# A Cooperative and Incentive-Based Proxy-and-Client Caching System for On-Demand Media Streaming

**IP Tak Shun**

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

©The Chinese University of Hong Kong
August 2005

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.

UNIVERSITY LIBRARY SYSTEM

香港中文大學
圖書館系統
18 NOV 2006
THE CHINESE UNIVERSITY OF HONG KONG
UNIVERSITY LIBRARY SYSTEM

# Abstract

Proxy caching is a key technique to reduce transmission cost for on-demand multimedia streaming. The effectiveness of current caching schemes, however, is limited by the insufficient storage space and weak cooperations among proxies and their clients, particularly considering the high bandwidth demands from media objects.

In this thesis, we propose COPACC, a cooperative proxy-and-client caching system that addresses the above deficiencies. This innovative approach combines the advantages of both proxy caching and peer-to-peer client communications. It leverages the client-side caching to amplify the aggregated cache space and rely on dedicated proxies to effectively coordinate the communications. We propose a comprehensive suite of distributed protocols to facilitate the interactions among different network entities in COPACC. It also realizes a smart and cost-effective cache indexing, searching, and verifying scheme. Furthermore, we develop an efficient cache allocation algorithm for distributing video segments among the proxies and clients. The algorithm not only minimizes the aggregated transmission cost of the whole system, but also accommodates heterogeneous computation and storage constraints of proxies and clients.

We also address the incentive issue of COPACC. That is, 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.

We have extensively evaluated the performance of the cooperative and incentive-based proxy-and-client caching system under various network and end-system configurations. The results demonstrate that it achieves remarkably lower transmission cost as compared to pure proxy-based caching with limited storage space. On the other hand, it is much more robust than a pure peer-to-peer communication system in the presence of node failures. Meanwhile, its computation and control overheads are both kept in low levels. Furthermore, with the incentive mechanism incorporated, the proxies have a strong incentive to collaborate in COPACC, and the optimal net profit and social welfare are achieved in the cooperative resource allocation games.

## 摘要

代理緩存(Proxy Caching)技術是降低隨選流式媒體傳輸的一種關鍵技術。但是，現今的緩存技術在應用於頻寬要求較高的流式媒體時，會遇到存儲空間不足和代理間缺乏合作的問題。

在本篇論文裡，我們提出一個名叫 COPACC 的系統，它利用代理伺服器和終端用戶的合作緩存技術來解決上述缺點。這種革新的方法結合了代理緩存和對等用戶通訊的優勢。它透過集合對等用戶所獻出的儲存空間來放大總計緩存空間，並且倚賴專用代理有效地協調網絡實體之間的通訊。我們提議一套分散式通訊協議來協調網絡中不同實體的交流。它實現了一個精明且具成本效益的緩存索引、搜尋和認證功能。再者，我們開發了一套有效率的運算方法把視像封包分發到代理和用戶的緩存上。該算法不僅使整個系統的總計傳輸減到最小，而且適用於不同代理和用戶的運算和儲存上限。

我們亦注意到 COPACC 中的誘因問題(Incentive Issue)。換句話說，在代理提供緩存空間給系統的背後究竟有什麼動機。為使代理參與 COPACC，我們建議一個收入回饋(Revenue−Rewarding)計畫，這個計畫會根據代理所獻出的資源多少來決定回饋。我們利用賽局理論模型(Game−Theoretic Model)來分析各代理對收入回饋計畫的相互作用。因這系統在非合作的情況下無法取得任何良好的特性，所以我們建議兩個合作方案來使系統達到最理想的狀態。在那個狀態下，所有網絡實體的利益是最大的。

我們在各種不同的網絡和終端系統配置下測試 COPACC 的性能。結果證明它比一般代理緩存系統所需要的傳輸費用較低。另一方面，它在用戶故障的情況下比一般對等緩存系統更穩定。同時，它的運算時間和額外開支都在於較低的水平。此外，在誘因機制(Incentive Mechanism)的帶動下，代理十分樂意參加 COPACC，從而令整個系統獲得最佳的利潤和大眾利益。

# Acknowledgement

I would like to thank my supervisors Prof. John C.S. Lui and Prof. JiangChuan Liu for their invaluable advices and guidance during my research study. I would also like to thank Prof. Michael R. Lyu for his suggestions in my research.

I am grateful to my friends, CJ, Hang, Hackker, Ken, Leo and Szeto, for their kind discussions and helps during my study. I also appreciate the help and fun all my friends gave me over these two years.

Finally, I would like to express my gratitude to my family and SzeKee for their support and love throughout my life.

This work is dedicated to my parents.

# Contents

**Abstract** **i**

**Acknowledgement** **iv**

**1 Introduction** **1**

1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1.1 Media Streaming . . . . . . . . . . . . . . . . . . . . . . 1

1.1.2 Incentive Mechanism . . . . . . . . . . . . . . . . . . . . 2

1.2 Cooperative and Incentive-based Proxy-and-Client Caching . 4

1.2.1 Cooperative Proxy-and-Client Caching . . . . . . . . . 4

1.2.2 Revenue-Rewarding Mechanism . . . . . . . . . . . . . 5

1.3 Thesis Contribution . . . . . . . . . . . . . . . . . . . . . . 6

1.4 Thesis Organization . . . . . . . . . . . . . . . . . . . . . . 7

**2 Related Work** **9**

2.1 Media Streaming . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.2 Incentive Mechanism . . . . . . . . . . . . . . . . . . . . . . . 11

2.3 Resource Pricing . . . . . . . . . . . . . . . . . . . . . . . . . 14

**3 Cooperative Proxy-and-Client Caching** **16**

3.1 Overview of the COPACC System . . . . . . . . . . . . . . . 16

3.2 Optimal Cache Allocation (CAP) . . . . . . . . . . . . . . . . 21

3.2.1 Single Proxy with Client Caching . . . . . . . . . . . . 21

3.2.2 Multiple Proxies with Client Caching . . . . . . . . . . 24
3.2.3 Cost Function with Suffix Multicast . . . . . . . . . . 26
3.3 Cooperative Proxy-Client Caching Protocol . . . . . . . . . . 28
3.3.1 Cache Allocation and Organization . . . . . . . . . . . 29
3.3.2 Cache Lookup and Retrieval . . . . . . . . . . . . . . 30
3.3.3 Client Access and Integrity Verification . . . . . . . . 30
3.4 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . 33
3.4.1 Effectiveness of Cooperative Proxy and Client Caching 34
3.4.2 Robustness . . . . . . . . . . . . . . . . . . . . . . . . 37
3.4.3 Scalability and Control Overhead . . . . . . . . . . . . 38
3.4.4 Sensitivity to Network Topologies . . . . . . . . . . . . 40

**4 Revenue-Rewarding Mechanism** **43**
4.1 System Model . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.1.1 System Overview . . . . . . . . . . . . . . . . . . . . . 44
4.1.2 System Formulation . . . . . . . . . . . . . . . . . . . 47
4.2 Resource Allocation Game . . . . . . . . . . . . . . . . . . . . 50
4.2.1 Non-Cooperative Game . . . . . . . . . . . . . . . . . 50
4.2.2 Profit Maximizing Game . . . . . . . . . . . . . . . . . 52
4.2.3 Utility Maximizing Game . . . . . . . . . . . . . . . . 61
4.3 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . 74
4.3.1 Convergence . . . . . . . . . . . . . . . . . . . . . . . 76
4.3.2 Participation Incentive . . . . . . . . . . . . . . . . . . 77
4.3.3 Cost effectiveness . . . . . . . . . . . . . . . . . . . . . 85

**5 Conclusion** **87**

**A NP-Hardness of the CAP problem** **90**

**B Optimality of the Greedy Algorithm** **92**

**Bibliography** **95**

# List of Figures

3.1 The cooperative proxy-and-client caching architecture. . . . 17

3.2 Illustration of different portions of a video stream. The prefix is to be cached by proxies, while the prefix-of-suffix by clients. 17

3.3 A logical view of multi-proxy with client caching. . . . . . . 23

3.4 Greedy prefix allocation . . . . . . . . . . . . . . . . . . . . . 28

3.5 Cache Lookup and Retrieval. . . . . . . . . . . . . . . . . . . 31

3.6 An illustration of the cache lookup and retrieve operations. . 32

3.7 Transmission cost as a function of the total proxy-client cache space. $S^p : S^c = 1 : 1$ . . . . . . . . . . . . . . . . . . . . . . 34

3.8 Transmission cost at different paths with suffix mulitcast. . . 35

3.9 Transmission cost versus the fraction of the proxy cache space in the total cache space. $r = S^p/(S^p + S^c) \times 100\%$ . . . . . . 36

3.10 Transmission cost versus client failure probability. . . . . . . 37

3.11 Transmission cost with different numbers of proxies and clients. 39

3.12 Control overhead with different number of proxies and streaming rates. . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

3.13 Transmission cost as a function of the number of proxies under real and synthetic network topologies. . . . . . . . . . . . . 41

4.1 The architecture of the cooperative proxy caching system. . . 45

4.2 Profit Maximizing Algorithm for the NSP in the Profit Maximizing Game. . . . . . . . . . . . . . . . . . . . . . . . . . . 58

4.3 A sample plot of total cache space $q$ versus price $p$. . . . . . . 59
4.4 A sample plot of net profit $E_p(p)$ versus price $p$. . . . . . . . 60
4.5 Mechanism of the cost minimization protocol. . . . . . . . . . . 68
4.6 Cost Maximization Protocol for the proxy coordinator. . . . . 70
4.7 A sample plot of minimum cost versus cache quantity $q$. . . . 72
4.8 A sample plot of social utility versus cache quantity $q$. . . . . 72
4.9 Cost and revenue functions being used in the evaluation. . . . 75
4.10 Quantity of cache supplied by each proxy. . . . . . . . . . . . 78
4.11 Net profit in each resource allocation game. . . . . . . . . . . 79
4.12 Net profit of the NSP in the system with $\theta'$ varied from 0.01 to 0.08. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
4.13 Net utility of each proxy. . . . . . . . . . . . . . . . . . . . . 81
4.14 Social utility in each resource allocation game. . . . . . . . . 82
4.15 Social utility in the system with $\theta'$ varied from 0.01 to 0.08. . 82
4.16 Sum of the social utility and the net profit in each resource allocation game with $\theta'$ varied from 0.01 to 0.08. . . . . . . 84

# List of Tables

3.1 Parameters of the COPACC system. . . . . . . . . . . . . . . . 20

4.1 A summary of the notations. . . . . . . . . . . . . . . . . . . . . 47

4.2 Parameters used in the resource allocation game. . . . . . . . 76

4.3 Cost per unit cache supplied by the proxies in different game. 85

# Chapter 1

# Introduction

Today's Internet has been increasingly used for carrying multimedia traffic, and on-demand streaming for clients of asynchronous playback requests is amongst the most popular networked media services. Given its broad spectrum of applications, like NetTV and distance learning, it has attracted much attention with many practical deployments in recent years [50]. The limited server capacity and the unpredictable Internet environment, however, make efficient and scalable on-demand media streaming remain a challenging task.

## 1.1 Background

### 1.1.1 Media Streaming

To reduce server/network loads, an effective means is to cache frequently used data at proxies close to clients [35, 47]. Streaming media, particularly those with asynchronous demands, could also benefit with a significant performance improvement from proxy caching given their static nature in content and highly localized access interests. However, media objects have high data rates and long playback durations, which combined yield a huge data volume. For illustration, a one-hour standard MPEG-1 video has a volume of about 675 MB; several such large streams will quickly exhaust the cache space of a standalone proxy. As such, it is necessary to design partial caching algorithms or group proxies to enlarge cache space [35, 12, 42, 51]. There have been extensive studies toward these directions, but the storage space of existing proxies are still far from satisfactory for media objects, and thus remains a bottleneck of the whole system.

Another approach is to generalize the proxy functionalities into every client [14, 26]. Such a peer-to-peer communication paradigm allows economical clients to contribute their local storage spaces for streaming. Specifically, the video data originally provided by a server are spread among clients of asynchronous demands, and each client can store the full or partial versions of the video stream in its local cache. Then, one or more clients can collectively supply cached data to other clients, thus amplifying the system capacity with increasing suppliers over time. However, in contrast to the reliable and dedicated servers or proxies, the loosely-coupled autonomous end-hosts can easily crash or leave without notice. Given that a media playback lasts a long time and consumes huge resources, a pure peer-to-peer system can be highly vulnerable in the Internet environment. As there are no authoritative parties, it is also difficult to identify and penalize malicious clients that intentionally inject forged data.

A hybrid caching system that combines the advantages of both proxy caching and peer-to-peer client communication can be used to address the above deficiencies. With the cooperation between the proxies and the clients in the network, the total network traffic of the media streaming can be significantly reduced.

### 1.1.2 Incentive Mechanism

The 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 [32] [7], collaborative web caching [49], P2P streaming [54], distributed computing [1], etc. It is generally agreed that the cooperative network performs significantly better than the traditional server-client 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 the peers (or network entities) 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 [3] 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 peers to cooperate is critical to the success of such systems.

To increase the involvement of network entities, participation incentive mechanisms [36] have to be used to effectively encourage them to collaborate in the network [41]. 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.

## 1.2 Cooperative and Incentive-based Proxy-and-Client Caching

In this thesis, we propose a cooperative and incentive-based proxy-and-client caching system for on-demand media streaming. The system consists of two components: *Cooperative Proxy-and-Client Caching (COPACC)* and *Revenue-Rewarding Mechanism*.

### 1.2.1 Cooperative Proxy-and-Client Caching

We propose a novel cooperative proxy-and-client caching system called COPACC. The innovative approach in COPACC combines the advantages of both proxy caching and peer-to-peer client communications. We leverage the client-side caching to amplify the aggregated cache space and rely on dedicated proxies to effectively coordinate the communications. We develop an efficient cache allocation algorithm that distributes video segments among the proxies and clients. The algorithm not only minimizes the aggregated transmission cost of the whole system, but also accommodates heterogeneous computation and storage constraints of proxies and clients. COPACC also makes effective use of multicast delivery in local regions, which further reduces the cost of the system.

COPACC also incorporates with a comprehensive suite of distributed protocols to facilitate the interactions among different network entities. Most operations in this protocol suite are executed by dedicated proxies. As such, it is not only suitable for clients with limited computation power, but also resilient to client failures. We also embed an efficient indexing and searching algorithm for video contents cached across different proxies or clients, as well as a signature verification mechanism, which can effectively identify and block malicious clients.

The performance of COPACC is extensively evaluated under various network and end-system configurations. The results demonstrate that it achieves remarkably lower transmission cost as compared to proxy-based caching with limited storage space. On the other hand, with the assistance from dedicated proxies, it is much more robust than a pure peer-to-peer system. Its transmission cost only slightly increases when a large portion of clients fail, even though the clients contribute a significant fraction in the total cache space. Moreover, it scales well to larger networks, and the cost generally reduces when more proxies and clients cooperate with each other.

## 1.2.2 Revenue-Rewarding Mechanism

We also propose a revenue-rewarding scheme to address the incentive issue. This incentive mechanism works complementary with COPACC 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 [39].

Our scheme follows the contribution reward-based incentive approach to reward the proxies by part of the transmission cost saved from COPACC. We focus on how proxies' 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. Both centralized and distributed algorithms are presented for the games to achieve different optimal situation.

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.

## 1.3 Thesis Contribution

The major contributions of the thesis are in two folds. First, we propose COPACC, a cooperative proxy-and-client caching system, to minimize the network transmission cost for media streaming. Second, we address the incentive issue of the COPACC by suggesting a revenue-rewarding mechanism. The contributions are summarized as follows:

**Cooperative Proxy-and-Client Caching System:**

- An efficient yet optimal cache allocation algorithm is proposed to distribute video segments among proxies and clients such that the aggregated network transmission cost is minimized.

- A comprehensive suite of distributed protocols are presented to facilitate the interactions among different entities in the cooperative network.

**Revenue-Rewarding Mechanism:**

- A revenue-rewarding scheme is proposed to address the incentive issue in COPACC. It provides a strong incentive for the network entities to contribute in the system.
- Game theoretic model is used to analyze the interactions among proxies under the revenue-rewarding scheme. It shows that no system-wide property is achieved in non-cooperative game.
- Two cooperative games are proposed to achieve different system-wide objectives. It shows that net profit and social welfare are maximized, and a cost-effective resource allocation is achieved in the cooperative games.

## 1.4 Thesis Organization

This thesis is organized as follows:

- Chapter 1 is an introduction of this thesis. It gives an overview of the background of this work. It also briefly describes the proposed cooperative and incentive-based proxy-and-client caching system. Moreover, it outlines the contribution and the organization of this thesis.
- Chapter 2 gives a literature review about Media streaming, Incentive mechanism and Resource pricing.
- Chapter 3 presents an overview of the COPACC architecture. It derives an efficient algorithm for cache allocation, and describes the cooperative caching protocol. It also evaluates COPACC with different performance metrics.

- Chapter 4 presents an incentive mechanism for COPACC. It gives an overview of the system, and presents the mathematical formulation. It 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 also evaluated.

- Finally, chapter 5 concludes the thesis.

□ **End of chapter.**

# Chapter 2

# Related Work

In this chapter, we review the works that are related to our proposed cooperative and incentive-based proxy-and-client caching system. Three kinds of work are presented here: Media streaming, Incentive mechanism and Resource pricing.

## 2.1 Media Streaming

### Proxy Caching for Media Streaming

Proxy caching for media streaming has attracted much attention in the past decade, and numerous algorithms have been proposed in the literature, e.g., run-length caching [13], prefix caching [42], and segment caching [12, 51, 38]; see a comprehensive survey in [35]. Considering the static nature of video contents and their intensive I/O demands, many of the algorithms employ a semi-static caching approach, where popular video portions are cached over a relatively long time period, rather than dynamically saved or replaced in response to individual client requests. COPACC also advocates semi-static caching, and its cache allocation is closely related to the prefix-suffix partition and stream segmentation algorithms [46]. However, these studies generally focus on a single proxy case with no cooperation among proxies.

It is well-recognized that proxies grouped together can achieve better performance than independent standalone proxies [16, 28]. An example for media caching is MiddleMan [2], which operates a collection of proxies as a scalable cache cluster; media objects are segmented into equal-sized segments and stored across multiple proxies, where they can be replaced at a granularity of a segment. There are also several local proxies responsible to answer client requests by locating and relaying the segments. To achieve better load balance and fault tolerance, a Silo data layout is suggested in [10], which partitions a media object into segments of increasing sizes, stores more copies for popular segments, and yet guarantees at least one copy stored for each segment. Our work is motivated by these cooperative systems, and we enhance them by combining proxy caching and client-side caching, which greatly expands the aggregated cache storage with contributions from the less expensive clients.

### P2P Media Streaming

Peer-to-peer communications have recently become a popular alternative to the traditional server-client paradigm. There are a series of pioneer works on peer-to-peer streaming, e.g., PROMISE [26], ZIGZAG [45], and CoopNet [40], which have demonstrated the superior scalability of shifting all functionalities to end-hosts. Yet, we are aware that, in contrast to the reliable and dedicated servers or proxies, the loosely-coupled autonomous end-hosts can easily crash, leave without notice, or even refuse to share its own data. Given that a media playback lasts a long time and consumes huge resources, we believe that dedicated proxies could still play an important role in building high-quality media streaming systems, as suggested in [11, 55]. Different from COPACC which focuses on caching, the key issue addressed in these studies is the optimal construction of an overlay structure. For storage allocation and management in a hybrid system, an optimal replication algorithm is proposed in [27], and a cooperative algorithm between a single proxy and its clients in a local area network is presented in [20]. COPACC complements them by considering a more general system with multiple cooperative proxies with client caching. A two-level hybrid architecture is exploited in [34], where an overlay network is used in the upper level to deliver videos from a central server to proxies and a collaborative-client network using loopback mechanism is applied in the lower level to transmit video data from proxy to clients. In loopback, cache is dynamically updated, which introduces an intensive disk I/O demand for the clients. Given that the video access pattern changes slowly, semi-static caching is adequate and it can be practically implemented. Moreover, Loopback concentrates on the collaboration between proxy and its clients only, but we also emphasize the importance of cooperative caching between proxies in reducing cost.

## 2.2 Incentive Mechanism

Recently a lot of efforts have been made to address the problems of *free-riding* and *tragedy of the commons* [24] 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

*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. A game theoretic framework has been suggested in [8] 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 [37], 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. The authors in [23] 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.

### Reputation-based Incentive

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 [32], 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 [21], 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, [52] 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. The authors in [17] 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 [30]. They used iterated prisoner's dilemma to model the peers' interaction, and proposed a reputation-based trust model with incentive mechanism incorporated.

### Contribution Reward-based Incentive

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. A micro-payment mechanism have been proposed in [19] 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 [44], 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 service provider to the proxies. We are aware of a similar work in [48], 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.

## 2.3 Resource Pricing

Our work relies on pricing the resource in order 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 [6], 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. In [9], the authors 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, [25] 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. The authors in [22] 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 proxy caching system, but suggest different pricing strategy to achieve different objective.

□ **End of chapter.**

# Chapter 3

# Cooperative Proxy-and-Client Caching

In this chapter, we present COPACC [29], a novel cooperative proxy-and-client caching system. We first give an overview of the COPACC system, and point out the key issues addressed in the COPACC architecture. Then, we present an efficient cache allocation algorithm as well as a comprehensive suite of cooperative caching protocols. Lastly, we evaluate the COPACC system with different performance metrics.

## 3.1 Overview of the COPACC System

Fig. 3.1 depicts a generic architecture of COPACC. A cluster of proxies are logically connected through direct or indirect peer links to form a proxy overlay, and each of them serves as the *home proxy* for a set of local clients. We assume that proxies and their clients are closely located with relatively low communication costs, e.g., they could be in the same ISP domain or in the same metropolitan area. A server storing the repository of videos, however, is far away from them, and the remote communications incur much higher costs.

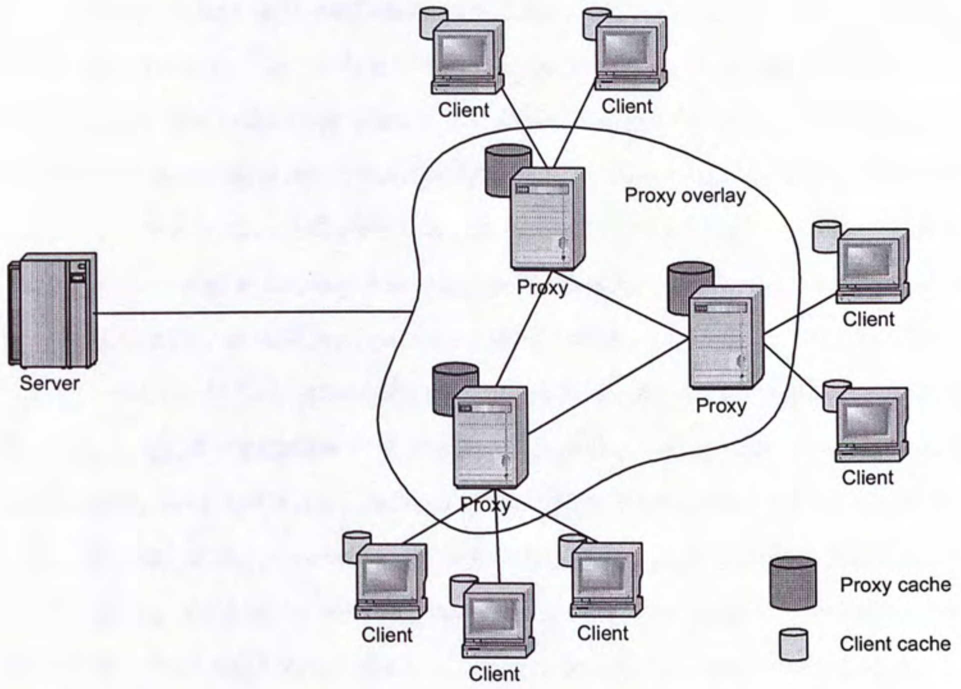


Figure 3.1: The cooperative proxy-and-client caching architecture.

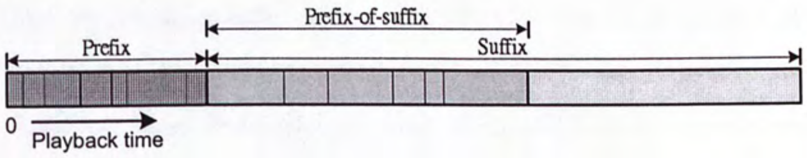


Figure 3.2: Illustration of different portions of a video stream. The prefix is to be cached by proxies, while the prefix-of-suffix by clients.

The video data are cached across both proxies and clients. We assume that the storage space of a proxy or a client is limited; the videos thus can be partially cached only, and there is always a full copy at the server. Specifically, as shown in Fig. 3.2, a video stream is partitioned into a *prefix* and a *suffix*, and the beginning part of the later is also referred to as the *prefix-of-suffix*. The proxies are responsible to cache the prefix of video, whereas the clients cache the prefix-of-suffix of video. This setting not only reduces the initial playback latency but also facilitates the multicast delivery with dynamic clients, as will be illustrated later. When a client expects to play a video, it first initiates a playback request to its home proxy, which intercepts the request and computes a streaming schedule: when and where to fetch which portion of the video. It then accordingly fetches the prefix, prefix-of-suffix, as well as the remaining part of suffix, and relays the incoming stream to the client. If needed, a proxy may also perform a verification operation, which detects forged video data through a simple signature mechanism.

Considering the video contents and their access patterns are relatively stable in several hours or even days, we advocate semi-static caching in COPACC. The cached contents are updated only when the system parameters have drastically changed, and a cache reconfiguration is then applied through a progressive cache filling mechanism.

There are two key issues to be addressed in the COPACC architecture:

- How to partition each video and allocate the prefixes and prefix-of-suffixes to different proxy and client caches? The objective is to minimize the total transmission cost of the COPACC system given the video access patterns, the heterogeneous transmission costs, and the storage constraints.

- How to manage, search, and retrieve the cached data in different proxies and clients? These operations should be highly efficient so as to deploy COPACC in large-scale networks with intensive requests.

To address the above challenges, we present an efficient allocation algorithm as well as a comprehensive suite of cache management and search protocols in the next two sections. Before proceeding our discussions, we first list the notations and parameters for COPACC, which are also summarized in Table 3.1.

We assume that there are $H$ cooperative proxies, indexed from 1 through $H$, and proxy $j$ serves as the home proxy for $K_j$ local clients. The video repository at the server includes $N$ Constant-Bit-Rate (CBR) videos, and video $i$ has length $L^i$ seconds and rate $b^i$ bps, $i = 1, 2, ..., N$. The total average access rate at proxy $j$ is $\lambda_j$, and the probability for accessing video $i$ is $f^i_j$ ($\sum_{i=0}^{N} f^i_j = 1$). We assume such statistics are known *a priori*, or obtained through online monitoring.

For cache allocation, there is a basic unit of $u$, also called *cache grain*, which is a hardware or operating system constraint, e.g., the size of a disk block. The cache space for proxy $j$ is $s^p_j$ units, and that for client $k$ of proxy $j$ is $s^c_{j,k}$ units. The volume of video $i$ is also represented as a number of units, i.e., $V^i = b^i L^i / u$ units. In practice, the aggregated cache space is less than the total volume of all the videos, i.e. $S^p + S^c \leq \sum_{i=1}^{N} V^i$, where $S^p = \sum_{j=1}^{H} s^p_j$ and $S^c = \sum_{j=1}^{H} \sum_{k=1}^{K_j} s^c_{j,k}$ are the total proxy cache size and total client cache size.

The cost for transmitting one unit of data from the server to a proxy is denoted by $w^{s \to p}$, and, similarly, the unit cost from proxy $j$ to proxy $k$ and that from proxy $j$ to its own clients are represented by $w^{p \to p}_{j,k}$ and $w^{c \leftrightarrow p}_j$, respectively.

We use $P^i$ to denote the prefix size (in units) of video $i$, and, $Q^i$, the prefix-of-suffix size. Both the prefix or prefix-of-suffix of a video are further partitioned into several segments and cached at a proxy or client. For video $i$, the size of a prefix segment cached in proxy $j$ is represented by $p^i_j$, and the size of a prefix-of-suffix segment cached at the client $k$ of proxy $j$ is $q^i_{j,k}$.

| **Parameter** | **Definition** |
|---|---|
| $N$ | Number of the videos. |
| $V^i$ | Volume of video $i$ (in units). |
| $H$ | Number of proxies. |
| $K$ | Number of clients. |
| $K_j$ | Number of local client attached to proxy $j$. |
| $s_j^p$ | Cache space of proxy $j$ (in units). |
| $S^p$ | Total cache space of all proxies (in units). |
| $s_{j,k}^c$ | Cache space of client $k$ of proxy $j$ (in units). |
| $S^c$ | Total cache space of all clients (in units). |
| $\lambda_j$ | Total access rate at proxy $j$. |
| $f_j^i$ | Probability for accessing video $i$ at proxy $j$. |
| $w^{s\to p}$ | Transmission cost per unit data from server to proxy. |
| $w_{j,k}^{p\to p}$ | Transmission cost per unit data from proxy $j$ to proxy $k$. |
| $w_j^{c\leftrightarrow p}$ | Transmission cost per unit data from proxy $j$ to its client. |
| $w^{in}$ | Internal cost per unit data of a proxy |
| $P^i$ | Prefix size of videos $i$ (in units). |
| $p_j^i$ | Size of the prefix segment of video $i$ cached in proxy $j$. |
| $Q^i$ | Prefix-of-suffix size of videos $i$ (in units). |
| $q_{j,k}^i$ | Size of the prefix-of-suffix segment of video $i$ cached at client $k$ of proxy $j$. |

Table 3.1: Parameters of the COPACC system.

The segment sizes are to be determined by the cache allocation algorithm, and the exact positions of the segments are to be determined by the cache organization protocol.

## 3.2 Optimal Cache Allocation (CAP)

The optimal cache allocation problem (CAP) in COPACC can be formulated as follows,

$$\begin{aligned}
&\mathbf{CAP}: \min Cost(\{p_j^i\}, \{q_{j,k}^i\}), \qquad (3.1)\\
&\text{s.t. } p_j^i, q_{j,k}^i \geq 0, j \in [1 \ldots H], k \in [1 \ldots K_j],\\
&\textstyle\sum_{i=1}^{N} p_j^i \leq s_j^p,\\
&\textstyle\sum_{i=1}^{N} q_{j,k}^i \leq s_{j,k}^c,\\
&\textstyle\sum_{j=1}^{H} p_j^i + \sum_{j=1}^{H} \sum_{k=1}^{K_j} q_{j,k}^i \leq V^i,
\end{aligned}$$

where $Cost(\{p_j^i\}, \{q_{j,k}^i\})$ is the function of the total transmission cost (per unit time) given allocation $\{p_j^i\}$ and $\{q_{j,k}^i\}$; the second and third constraints follow the cache space limit of proxy $j$ and that of client $k$ of proxy $j$, respectively; the forth constraint applies because we do not consider replication in this study. In this section, we start our discussion from a simple scenario of no cooperation between proxies, where the cache allocation for each proxy and its own clients can be examined independently. We derive an efficient optimal solution for this scenario, which is then extended to accommodate multiple cooperative proxies with client caching, i.e., a general COPACC system.

### 3.2.1 Single Proxy with Client Caching

As said, we focus on a single proxy and its clients, both of which contribute cache spaces, but there is no interactions with other proxies nor their clients.

Since the transmission costs between this proxy and all its clients are identical, we refer to this system as a homogeneous cost system. We drop the proxy index (subscript $j$) from the relevant parameters for ease of exposition.

This homogeneous cost system has a nice property that the total transmission cost depends only on how the video streams are partitioned into prefixes and prefix-of-suffixes for caching. This is because all prefixes are to be cached at the single proxy, and any allocation of the prefix-of-suffix segments across the local clients yield the same cost due to the uniform cost for proxy-client transmissions. As such, we can combine the cache of all the clients to form an aggregated cache space $S^c$, and, to derive the minimum transmission cost, we only need to find the optimal values of $\{P^i\}$ and $\{Q^i\}$ subject to cache space constraints $S^p$ and $S^c$.

We define an auxiliary cost function $C^i(P^i, Q^i)$, which is the cost for delivering video $i$ with prefix size $P^i$ and prefix-of-suffix size $Q^i$. Note that $Cost(\{p^i_j\}, \{q^i_{j,k}\})$ is now equal to $\sum_{i=1}^{N} C^i(P^i, Q^i)$ in this simple scenario. Moreover, minimizing it is equivalent to maximizing the cost saving against the system with no caching, i.e. maximizing $\sum_{i=1}^{N}[C^i(0,0) - C^i(P^i, Q^i)]$.

We use a dynamic programming approach to solve the problem. Let $B$ be a three-dimensional matrix, where $B(i, t^p, t^c)$ represents the maximum cost saving for videos 1 through $i$ $(1 \le i \le N)$, when $t^p$ $(0 \le t^p \le S^p)$ units of proxy cache and $t^c$ $(0 \le t^c \le S^c)$ units of client cache are used. We have

$$B(i, t^p, t^c) = \begin{cases} 0, \quad i = 0,\ 0 \le t^p \le S^p,\ 0 \le t^c \le S^c \\ \max\{B(i-1, t^p - v^p, t^c - v^c) + C^i(0,0) - C^i(v^p, v^c)\}, \\ \qquad\qquad 0 \le v^p \le t^p, 0 \le v^c \le t^c, v^p + v^c \le V^i. \end{cases}$$

The matrix can be filled in plane-order starting from $B(0,0,0)$ to $B(N, S^p, S^c)$, and the latter gives the maximum cost saving. The minimum total transmission cost is therefore $\sum_{i=1}^{N} C^i(0,0) - B(N, S^p, S^c)$, and the corresponding prefix and prefix-of-suffix partitioning can be obtained through backtracking the iterations.

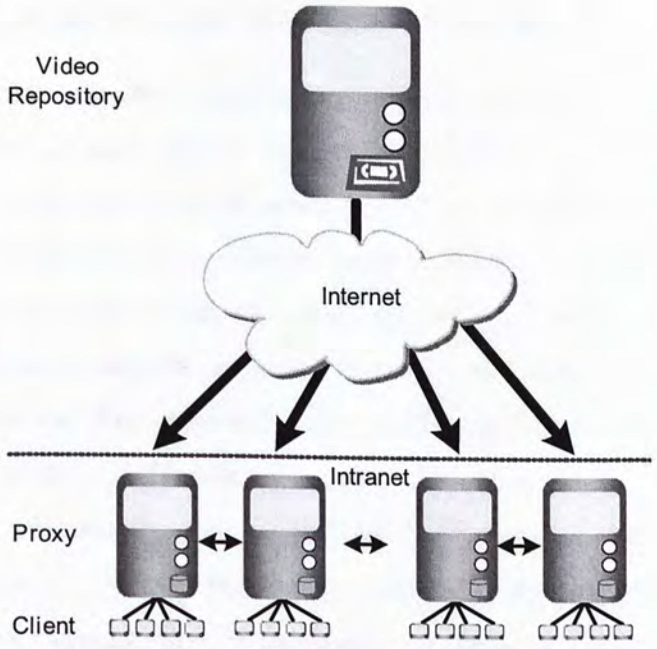


Figure 3.3: A logical view of multi-proxy with client caching.

This dynamic programming algorithm has time complexity $O(N \cdot S^p \cdot S^c \cdot M)$, where $M = \max_{1 \le i \le N}(1+V^i)V^i/2$. It is applicable with arbitrary cost function $C^i(P^i, Q^i)$, which can be instantiated given a specific transmission scheme. As an example, assume both a server-to-client and a client-to-client transmissions are unicast-based and relayed by a proxy, $C^i(P^i, Q^i)$ can be derived as $\lambda f^i \cdot [w^{c \leftrightarrow p} P^i + 2w^{c \leftrightarrow p} Q^i + (w^{s \to p} + w^{c \leftrightarrow p})(V^i - P^i - Q^i) + w^{in}(P^i + Q^i)]$, where the first four terms in the second part respectively represent the costs for retrieving prefix, prefix-of-suffix, the remaining suffix, and the internal cost of the proxy, for each playback request. Note that $\lambda$ is the total access rate of the proxy, and $w^{in}$ is the internal cost per unit data handled by the proxy. When there is no caching ($P^i = Q^i = 0$), we have $C^i(0,0) = V^i \lambda f^i \cdot (w^{s \to p} + w^{c \leftrightarrow p})$.

In the end of this section, we further introduce multicast delivery to the system and derive the corresponding cost function.

### 3.2.2 Multiple Proxies with Client Caching

We now consider the case of multiple cooperative proxies with client caching. Fig. 3.3 offers a logical view of this general COPACC system, in which the segment of prefix and prefix-of-suffix of a video are placed across different proxies and their clients, respectively, and the transmission of a video stream thus involve interactions among several proxies and clients. Moreover, the unit transmission costs for the proxy-to-proxy and client-to-proxy links can be heterogeneous. The cache allocation problem (CAP) thus becomes much more complex than in the homogeneous cost system.

In fact, we formally prove that CAP is NP-hard in this general case (see *Appendix A*). We thus resort to a practically efficient heuristics, which consists of two phases: first, it partitions the prefix and prefix-of-suffix for each video; second, given the partitions, it allocates the segments of prefixes and prefix-of-suffixes to the proxies and clients.

**1) Partitioning of prefix and prefix-of-suffix:** In this phase, we calculate the optimal values of $P^i$ and $Q^i$ for each video, and, to achieve a computationally efficient solution, we do not address their allocation across the proxies and clients. Instead, we approximate the system by a single proxy system with aggregated proxy cache space $S^p$ and aggregated client cache space $S^c$. Other parameters are approximated as follows: video access rate $\lambda = \sum_{j=1}^{H}\lambda_j$, access probability $f^i = (1/\lambda)\sum_{j=1}^{H}\lambda_j f_j^i$, unit transmission cost $w^{c\leftrightarrow p} = (1/K)\sum_{j=1}^{H}K_j w_j^{c\leftrightarrow p}$, and internal cost $w^{in} = (1/H^2)\sum_{j=1}^{H}\sum_{k=1}^{H} w_{j,k}^{p\rightarrow p}$, that is, we consider the cost for proxy-to-proxy transmissions as an internal cost, and assume $w_{j,k}^{p\rightarrow p}$ is 0 if $j = k$.

Given the above transformation, an approximate solution can be directly obtained using the dynamic programming algorithm for the homogeneous cost system.

**2) Allocation to proxy and client caches:** In this phase, we allocate the prefix and prefix-of-suffix to the proxies and clients so as to meet the storage constraints at each proxy and client. Since given $P^i$ and $Q^i$ obtained in the first phase, the allocation for prefixes to proxy caches is independent from that for prefix-of-suffixes to client caches, and vice versa, we separate the two allocation problems and solve them individually.

We first consider the allocation for prefixes. Let $W^p(i,j,p^i_j)$ be the transmission cost when the segment of size $p^i_j$ from the prefix of video $i$ is stored in proxy $j$. The problem for optimal prefix allocation is then formulated as

$$\mathbf{PA}: \min \sum_{i=1}^{N} \sum_{j=1}^{H} W^p(i,j,p^i_j) \tag{3.2}$$

$$s.t. \quad \sum_{j=1}^{H} p^i_j = P^i, \quad i \in [1 \dots N]$$

$$\sum_{i=1}^{N} p^i_j \leq s^p_j, \quad j \in [1 \dots H].$$

For unicast delivery, $W^p(i,j,p^i_j)$ can be instantiated as

$$W^p(i,j,p^i_j) = \sum_{j'=1}^{H} p^i_j \left[ w^{p \to p}_{j,j'} + w^{c \leftrightarrow p}_{j'} \right] \lambda_{j'} f^i_{j'}. \tag{3.3}$$

Let $\bar{W}^p(i,j) = \sum_{j'=1}^{H} \left[ w^{p \to p}_{j,j'} + w^{c \leftrightarrow p}_{j'} \right] \lambda_{j'} f^i_{j'}$, the optimization objective for problem **PA** can be re-written as $\min \sum_{i=1}^{N} \sum_{j=1}^{H} \bar{W}^p(i,j) \cdot p^i_j$. Note that, $\bar{W}^p(i,j)$ is independent of $p^i_j$, and can be viewed as the transmission cost when each unit prefix data of video $i$ cached in proxy $j$. The above formulation for **PA** thus can be relaxed as a linear programming problem if $p^i_j$ is not restricted to integers. In practice, this is generally viable, for a video stream that can be partitioned with fine-granularity, and the total data cached in any proxy is less than its maximum capacity for any optimal solution to the linear programming.

Similarly, we can formulate the optimal allocation problem for prefix-of-suffixes to be cached at clients as follows,

$$\mathbf{SA}: \min \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{K_j} W^c(i,j,k,q^i_{j,k}) \tag{3.4}$$

$$s.t. \quad \sum_{j=1}^{H} \sum_{k=1}^{K_j} q^i_{j,k} = Q^i, \quad i \in [1 \dots N]$$

$$\sum_{i=1}^{N} q^i_{j,k} \leq s^c_{j,k}, \quad j \in [1 \dots H], k \in [1 \dots K_j]$$

where $W^c(i,j,k,q^i_{j,k})$ is the transmission cost when the segment of size $q^i_{j,k}$ from the prefix-of-suffix of video $i$ is stored in client $k$ of proxy $j$. For unicast delivery, $W^c(i,j,k,q^i_{j,k})$ is given by

$$\sum_{j'=1}^{H} q^i_{j,k}[w^{p\to p}_{j,j'} + w^{c\leftrightarrow p}_{j'} + w^{c\leftrightarrow p}_{j}]\lambda_{j'} f^i_{j'}, \tag{3.5}$$

which can be re-written as $\bar{W}^c(i,j,k) \cdot q^k_{i,j}$ if we define

$$\bar{W}^c(i,j,k) = \sum_{j'=1}^{H} \left[w^{p\to p}_{j,j'} + w^{c\leftrightarrow p}_{j'} + w^{c\leftrightarrow p}_{j}\right] \lambda_{j'} f^i_{j'}. \tag{3.6}$$

Obviously, both the cost function and the problem **SA** itself have similar structure as that of problem **PA**. The linear programming relaxation thus also applies.

We will show later that such relaxation also holds for multicast delivery.

### 3.2.3 Cost Function with Suffix Multicast

So far we have focused on unicast delivery only, and presented the corresponding cost functions. In this subsection, we further consider multicast delivery, which is known as an efficient vehicle for streaming to clients with requests close in time [46, 4]. However, though IP multicast has been widely adopted within ISP networks, its deployment over the global Internet remains confined. We thus assume multicast delivery at the path from a proxy to its local clients, but only unicast delivery from the server to a proxy or between two proxies.

Even though multicast is only enabled at local paths, a proxy can still serve a series of requests from its local clients for the same video using a *suffix batching* technique. Specifically, assume the first request for video $i$ arrives at time 0, the home proxy will fetch and relay the prefix of the video to this client through unicast, which takes $P^i u/b^i$ seconds; all the local requests arrive during interval $[0, P^i u/b^i]$ will then be batched with a single copy of the suffix for video $i$ being multicast to all the requested clients. In other words, the batching window is of size $P^i u/b^i$.

We now derive the cost function $C^i(P^i, Q^i)$ for the case of single-proxy with client caching. We assume that the video accesses follow a Poisson arrival, that is, the average number of requests arrived in the batching window for video $i$ is $1 + (P^i u/b^i)(\lambda f^i)$. The cost per request for multicasting the suffix to batch of clients is thus $[w^{c\to p}(V^i - P^i) + w^{c\to p}Q^i + w^{s\to p}(V^i - P^i - Q^i) + w^{in}Q^i]/[1 + (P^i u/b^i)(\lambda f^i)]$. Since a prefix is always delivered using unicast, the cost function $C^i(P^i, Q^i)$ is then given by:

$$\lambda f^i \cdot \{\frac{w^{c\to p}(V^i - P^i) + w^{c\to p}Q^i + w^{s\to p}(V^i - P^i - Q^i) + w^{in}Q^i}{1 + (P^i u/b^i)(\lambda f^i)} + (w^{c\to p} + w^{in})P^i\}. \tag{3.7}$$

Similarly, we can derive the cost function $\bar{W}^c(i,j,k)$ of problem **SA**. For a batching windows contains $1 + (P^i u/b^i)(\lambda_{j'} f^i_{j'})$ requests from proxy $j'$, we need only a single retrieval for the suffix distributed at client caches and the server. The cost function $\bar{W}^c(i,j,k)$ at proxy $j$ is thus

$$\sum_{j'=1}^{H} \left[\frac{w_j^{c\to p} + w_{j,j'}^{p\to p} + w_{j'}^{c\to p}}{1 + (P^i u/b^i)(\lambda_{j'} f^i_{j'})}\right] \lambda_{j'} f^i_{j'}. \tag{3.8}$$

Regarding the cost function $\bar{W}^p(i,j)$ of problem **PA**, it is exactly the same as that for unicast case because a prefix is delivery through unicast only. In addition, if $f^i_1 = f^i_2 = ... = f^i_H$, we have the following observations for $\bar{W}^p(i,j)$:

- Given $i, i' \in [1 \ldots N]$, $\bar{W}^p(i,j)/\bar{W}^p(i',j)$ is a constant for any $j \in [1 \ldots H]$;
- Given $j, j' \in [1 \ldots H]$, $\bar{W}^p(i,j)/\bar{W}^p(i,j')$ is a constant for any $i \in [1 \ldots N]$.

Since clients often have common interests, it is likely that the distributions of video access probabilities are similar at different proxies, that is, $f^i_1 = f^i_2 = ... = f^i_H$ holds. The above observation thus leads to an simpler yet optimal greedy algorithm for problem **PA**, as shown in Fig. 3.4.

```
1: Sort proxies in ascending order of cost $\bar{W}^p(1,j)$;
   Store the results in $j$-List;
2: Sort videos in descending order of cost $\bar{W}^p(i,1)$;
   Store the results in $i$-List;
3: $j^* \leftarrow$ first component of $j$-List;
   $i^* \leftarrow$ first component of $i$-List;
4: Cache as many units as possible for the prefix of video $i^*$ to proxy
   $j^*$;
5: If proxy $j^*$ has cache space left, then $i^* \leftarrow$ next component of $i$-List;
6: If prefix of video $i^*$ has not been fully cached, then $j^* \leftarrow$ next com-
   ponent of $j$-List;
7: Repeat steps 4 to 6 until all prefixes are allocated.
```

Figure 3.4: Greedy prefix allocation

Intuitively, this algorithm always cache the most expensive prefix into the cheapest proxy, so as to minimize the total transmission cost. Its complexity is $O(NlogN)$, which is generally lower than directly solving the linear programming problems (even if the simplex method [15] is used). A formal proof of the optimality of this greedy algorithm can be found in *Appendix B*.

## 3.3 Cooperative Proxy-Client Caching Protocol

As shown in Fig. 3.1, COPACC operates as a two-level overlay, where the first level consists of all the proxies, and the second level consists of each proxy and its own clients. The interactions among different entities in this two-level overlay are specified by a cooperative proxy-client caching protocol, which consists of three subprotocols: *cache allocation and organization*, *cache lookup and retrieval*, and *client access and integrity verification*. We now detail the operations, and address the practical issues toward realizing the COPACC system.

### 3.3.1 Cache Allocation and Organization

All the cache allocation and organization decisions are implemented in proxies. The protocol starts by establishing connections among the proxies, and an election algorithm is then executed to choose a coordinator. We currently employ the distributed Bully algorithm [18], which opts for the proxy of the highest computational power as the coordinator. The coordinator is responsible for collecting parameters from all other proxies and then running the optimal cache allocation algorithm described in the previous section.

Given $P^i = \sum_{j=1}^{H} p_j^i$ and $Q^i = \sum_{j=1}^{H} \sum_{k=1}^{K_j} q_{j,k}^i$, the interval of the prefix in video stream $i$ is simply $[0, P^i u/b^i]$ and that of prefix-of-suffix is $[P^i u/b^i, Q^i u/b^i]$. The coordinator should then determine the position of each segment to be allocated to proxies and clients in the prefix and prefix-of-suffix. Since the total transmission cost depends only on the segment size, COPACC employs a simple organization scheme: for prefix of video $i$, allocate segment of interval $[\sum_{m=1}^{j-1} p_m^i u/b^i, \sum_{m=1}^{j} p_m^i u/b^i]$ in the video stream to proxy $j$, and, for the prefix-of-suffix, allocate interval $[\sum_{m=1}^{j-1} \sum_{n=1}^{K_{j-1}} q_{m,n}^i u/b^i, \sum_{m=1}^{j} \sum_{n=1}^{K_j} q_{m,n}^i u/b^i]$ to the clients of proxy $j$, which further partitions this interval into segments to be cached in its local clients according to their cache spaces. Hence, the cache location of each interval of the stream can be easily calculated from $\{p_j^i\}$ and $\{q_{j,k}^i\}$. As the coordinator keeps a full copy of the allocations, a lookup request for the cache locations of a particular video stream can always be accomplished by contacting the coordinator. To balance the load of the proxies, the coordinator also distributes the lookup information uniformly to other proxies using a hash function $h(i)$; that is, for video $i$, a copy of its cache location information are kept by proxy $h(i)$ as well. Since the proxies are persistent and reliable nodes, even the simplest hashing like $h(i) = (i \ mod \ H) + 1$ will work well in COPACC. In other words, COPACC does not have to rely on a flooding-based search, nor a complex and costly distributed hash table (DHT), as in many peer-to-peer systems.

### 3.3.2 Cache Lookup and Retrieval

For each playback request for video $i$ from a client, its home proxy discovers and retrieves the video data on behalf of its clients. This is accomplished by first issuing a ***cache lookup*** request $R_{lookup}[i]$, which, according to the cache organization, can be directly submitted to proxy $h(i)$. Upon receiving the location information from proxy $h(i)$, the initiated proxy then issues a series of cache retrieval requests, $R_{retrieval}[i]$, to corresponding proxies for retrieving and then relaying the segments cached at proxies or their clients. Finally, the un-cached part of the suffix is retrieved from the server.

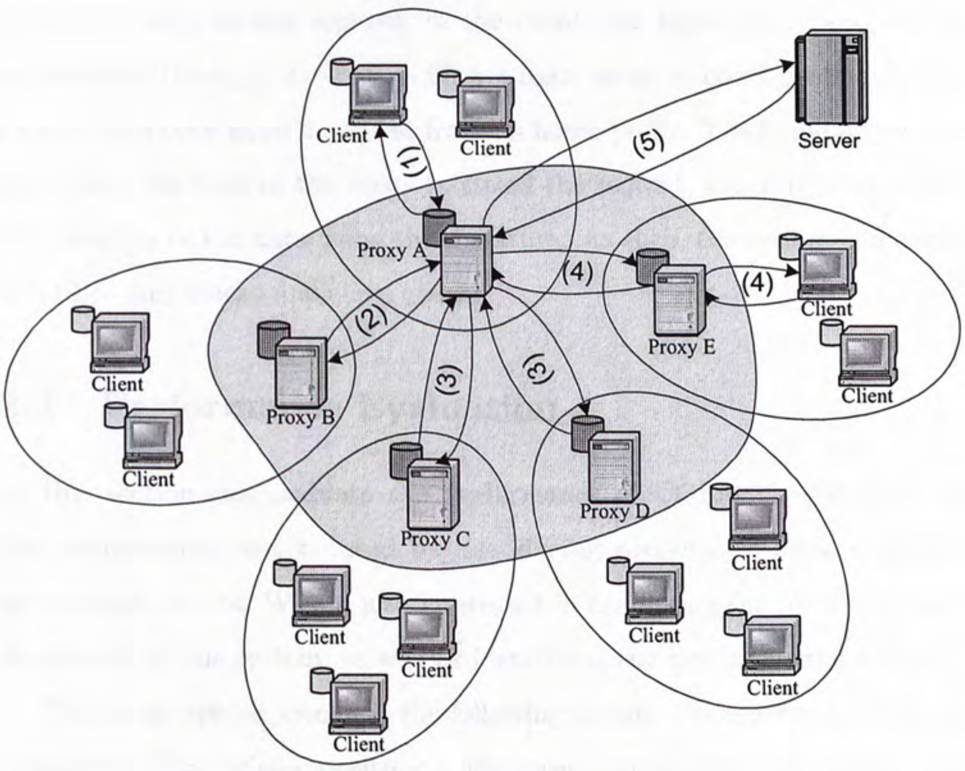
When a proxy receives a retrieval request, it first checks whether the requested data has been cached. If cached, it will stream the data to the requested proxy; if not, it will retrieve the data from the server, stores a copy in its own cache or its clients' cache, depending on whether the content belongs to a prefix or to a prefix-of-suffix, and then stream to the initiated proxy. This leads to a passive filling scheme with no need for a synchronized global replacement: the cache space are initially empty or represents an outdated allocation scheme; it is then filled up gradually following the requests from other proxies, which represents the updated allocation. An illustration of the steps for cache lookup and retrieval can be found in Fig. 3.5 and Fig. 3.6.

### 3.3.3 Client Access and Integrity Verification

The client-side operations are relatively simple, which can be easily implemented in economical but less powerful personal computers. In particular,

```
1: while Receive a request do
2:   if $R_{retrieval}[i]$ from local client then
3:     Look up proxy $j = h(i)$; get $\{p_j^i\}$ and $\{q_j^i\}$
4:     Send $R_{retrieval}[i]$ to proxy $j$ for prefix of interval
       $[\sum_{m=1}^{j-1} p_m^i u/b^i, \sum_{m=1}^{j} p_m^i u/b^i]$
5:     Send $R_{retrieval}[i]$ to proxy $j$ for prefix-of-suffix of interval
       $[\sum_{m=1}^{j-1} q_m^i u/b^i, \sum_{m=1}^{j} q_m^i u/b^i]$
6:     Retrieval remaining interval $[P^i + Q^i, L^i]$ from server
7:     Relay the stream to the request client
8:   else if $R_{retrieval}[i]$ for prefix of interval $[a, b]$ then
9:     Prefix of interval $[a, b]$ not exist in proxy cache $\rightarrow$ retrieval from
       server and store in proxy cache
10:    Send prefix of interval $[a, b]$ to requested proxy
11:  else if $R_{retrieval}[i]$ for prefix-of-suffix of interval $[a, b]$ then
12:    Prefix of interval $[a, b]$ not exist in the cache of any local client
       $\rightarrow$ retrieval from server and store in a local client's cache
13:    Send prefix-of-suffix of interval $[a, b]$ to requested proxy
14:  else if $R_{lookup}[i]$ from another proxy then
15:    Reply $\{p_j^i\}$ and $\{q_j^i\}$
16:  end if
17: end while
```

Figure 3.5: Cache Lookup and Retrieval.



(1) client request to home proxy for video $i$;

(2) location lookup request to proxy $B = h(i)$;

(3) retrieve and relay prefix segments from proxy cache;

(4) retrieve and relay prefix-of-suffix segments from clients;

(5) retrieve and relay the remaining part of suffix from server.

Figure 3.6: An illustration of the cache lookup and retrieve operations.

a client is not involved in managing the overlay, nor determining cache allocation and organization. It simply reports its available spaces to its home proxy. The home proxy then determines and keeps the location for data cached in its local clients, and then instructs the clients for caching the data. For each cached segment in the client, the home proxy also save a signature of the copy, such as its SHA-1 hash value. A client contributes its cached data only upon a request from its home proxy. The home proxy will then relay the data to the proxy initiated the request, and if needed, verify the integrity of the data using the signature. As such, the system can easily identifies and blocks malicious clients.

## 3.4 Performance Evaluation

In this section, we evaluate the performance of COPACC. We focus on the transmission cost reduced by introducing cooperative caching among proxies and clients. We are also interested in examining the robustness and scalability of this system, as well as identifying the key influential factors.

Unless otherwise specified, the following default settings are used in our evaluation. The video repository in the sever contains 100 CBR videos each of 512 Kbps rate. Their lengths are uniformly distributed in between 100 and 140 minutes; the mean (120 minutes) is a typical length of a movie. As suggested by existing studies on media access patterns, we assume the access probabilities of the videos follow a Zipf distribution with skew factor $\theta = 0.271$ [4]. The cache grain (unit) is set to the size of 2-minute video data. All the cache sizes discussed in this section are normalized by the total size of the video repository, and the transmission costs are normalized by the corresponding cost of a system with no cache. Therefore, our conclusions are also applicable to systems with proportionally scaled parameters.

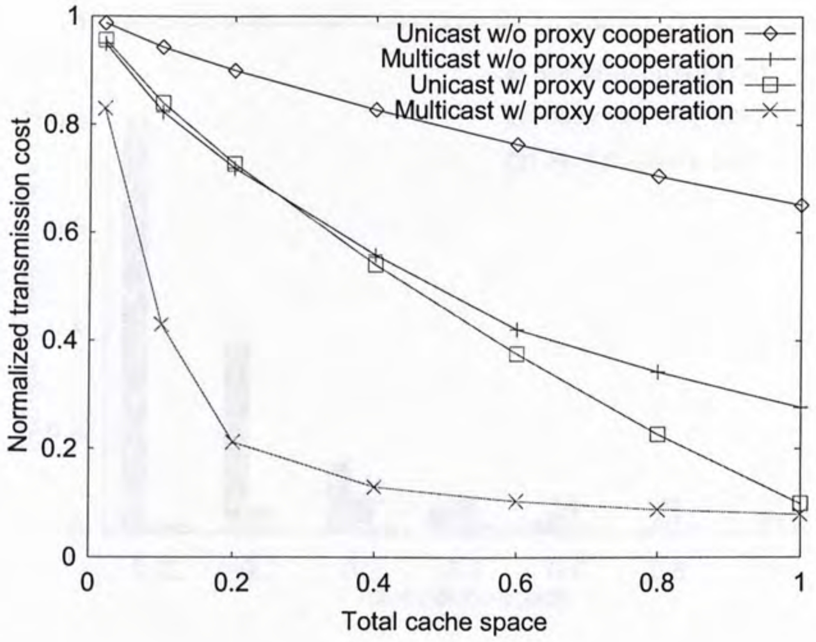


Figure 3.7: Transmission cost as a function of the total proxy-client cache space. $S^p : S^c = 1 : 1$

## 3.4.1 Effectiveness of Cooperative Proxy and Client Caching

A primary design objective of COPACC is to reduce the transmission cost for streaming to clients of asynchronous requests. Hence, in the first set of experiments, we examine the cost reduction under various proxy and client configurations.

We assume there are 4 proxies cooperated with each other, and the client access rate at each proxy is 50 requests per minutes. The ratio between the unit transmission costs of different paths is set to $w^{s\rightarrow p} : w^{p\rightarrow p} : w^{c\leftrightarrow p} = 10 : 3 : 1$. Note that, this setting is indeed conservative as compared to that in many previous studies [46]. In addition, we are interested in the normalized transmission cost, which depends on this ratio, while not the exact value at each path.

Fig. 3.7 plots the transmission cost as a function of the total cache space in the system, where $S^p : S^c = 1 : 1$, i.e., the proxies and clients respectively

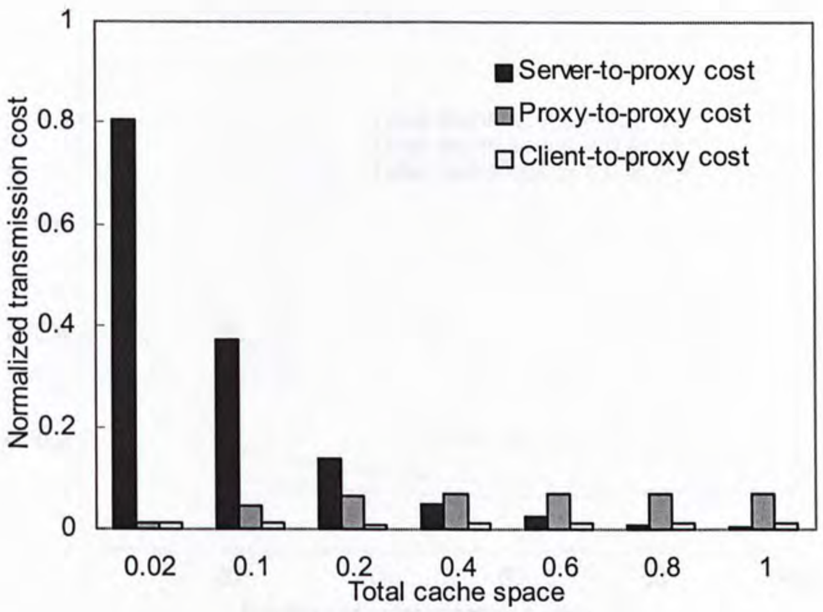


Figure 3.8: Transmission cost at different paths with suffix mulitcast.

contribute half of the total cache size. Not surprisingly, increasing the total space reduces transmission cost. With unicast, the cost decreases linearly, while with suffix multicast, it decreases much faster. When the total cache space is 0.2 (20% of the video repository), the cost with suffix multicast has been reduced to 0.2; in other words, a 20% cache space leads to a 80% cost reduction, which implies that batching the requests from local clients can avoid a significant amount of remote transmissions (server-to-proxy). This can also be verified by Fig. 3.8, which shows the cost due to server-to-proxy transmissions quickly decreases with an increase of the cache space, and becomes a minor part in the total transmission cost when the cache space is over 0.4.

In Fig. 3.7, we also show the cost when a proxy cooperates with its clients only, while not with other proxies. Clearly, the cost with cooperative proxies are much lower, particularly when multicast is also enabled in local paths. As such, in the following discussions, we focus on the results with

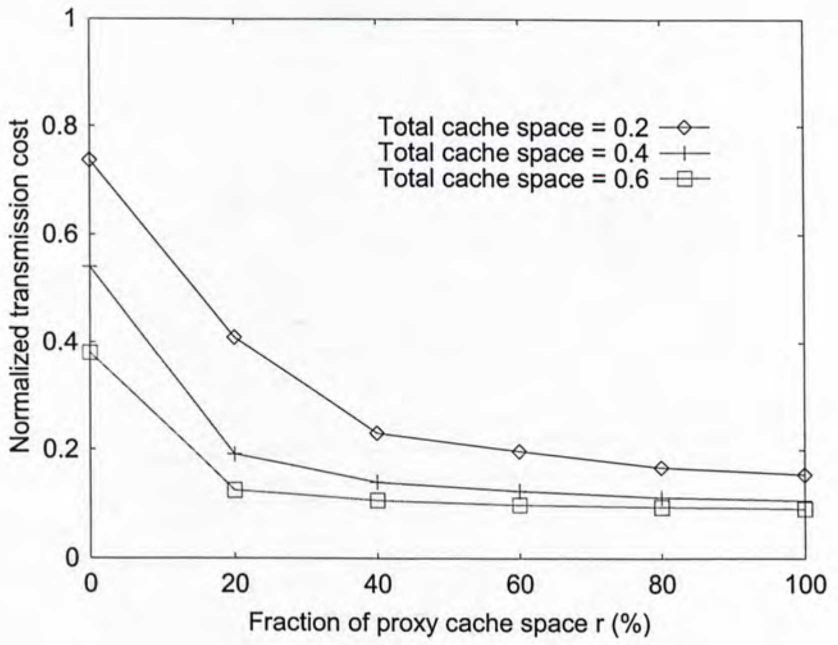


Figure 3.9: Transmission cost versus the fraction of the proxy cache space in the total cache space. $r = S^p/(S^p + S^c) \times 100\%$

cooperative proxies and multicast delivery only.

To further identify the respective contributions of proxy caching and client caching, Fig. 3.9 depicts the transmission cost versus the fraction of the proxy cache space in the total cache space. We can see that the transmission cost reduces when the proxies contribute a higher fraction in the total cache space of the system. Intuitively, the more cache space contributed by proxies, the more direct transmissions among proxies for delivering a video stream, which generally incur lower costs, because the video data fetched from a client's cache have to be relayed by proxies as well. The best performance is thus achieved when all cache space is in the proxies. Nonetheless, it is often expensive to upgrade dedicated proxies and add more disk spaces. On the other hand, from Fig. 3.9, we find that, even if the proxy caches constitute a small part in the total cache space, a near optimal cost can still be achieved. As an example, when the total cache space is 0.6 and only

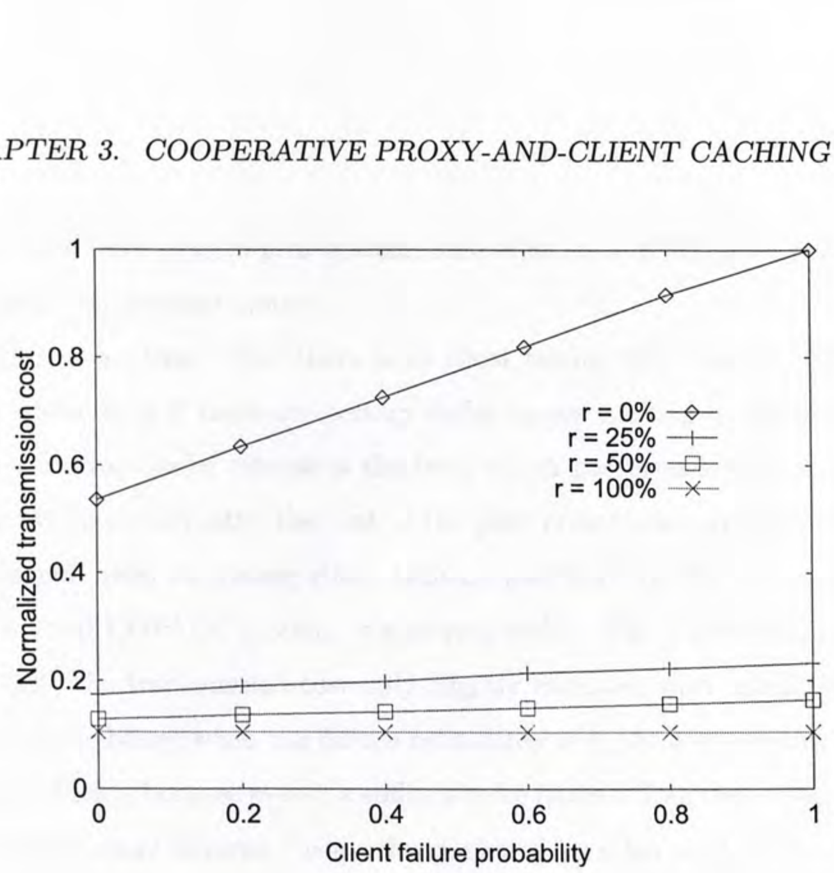


Figure 3.10: Transmission cost versus client failure probability.

20% is from proxies, i.e., the total proxy cache space is only 0.12, the cost is already less than 0.13, which is quite close to the optimal value (around 0.1) when the fraction of proxy cache is 100%. In other words, client caching well complements proxy caching, making COPACC a very economical alternative to pure proxy caching.

### 3.4.2 Robustness

As in peer-to-peer streaming systems, the robustness in the presence of client failures is also a critical concern in COPACC. To evaluate this, we assume that each client has certain failure probability when its own cache is accessed, but the video access rate from all clients remains constant. In Fig. 3.10, we show the transmission cost as a function of different client failure probabilities. The total cache space of the system is 0.4, and we vary, $r$, the fraction of the total proxy cache space in the total cache space from 0% to 100%, which represents two extreme cases: when $r = 0\%$, COPACC degenerates to a pure peer-to-peer system, and, when $r = 100\%$, it degenerates to a pure proxy-based system.

We can see that, when there is no client failure, the costs for different $r$ are quite close if there are certain cache spaces existing in proxies, and the pure proxy-based scheme is the best, which has been explained previously. More importantly, the cost of the pure proxy-based system remains unchanged when increasing client failures, and that for $0\% < r < 100\%$, or a normal COPACC system, is also very stable. For illustration, even if $r$ is 25%, the transmission cost only slightly increases with an increase of failure probability; when the failure probability is 1, the cost remains a low as 0.22. This is because even if a suffix is to be fetched from the server in the presence of client failures, the overhead, shared by a batch of clients, is not excessive. To the contrary, the cost of the pure peer-to-peer system quickly increases and reaches 1 (the cost of a zero-cache system), when all clients fail. Such results demonstrate that the use of dedicated proxies with suffix batching remarkably improves the robustness and resilience of COPACC in the presence of client failures, even if the total proxy cache space is minor as compared to the total client cache space.

## 3.4.3 Scalability and Control Overhead

We further explore the scalability of COPACC with larger number of proxies and clients. Fig. 3.11 shows the total transmission costs for different number of proxies and clients. In this set of experiments, the cache space of each proxy, $s_j^p$, is set to 0.03, and that of each client, $s_{j,k}^c$, is 0.005. The access rate from each client is set to 0.01 per minute. In other words, while a client joining the system contributes certain cache spaces, it also introduces more requests. Yet, we observe that the transmission cost slightly decreases with more clients, implying that client caching overcomes the increased loads. Note that the normalized cache space of each client is only 0.005, or equiv-

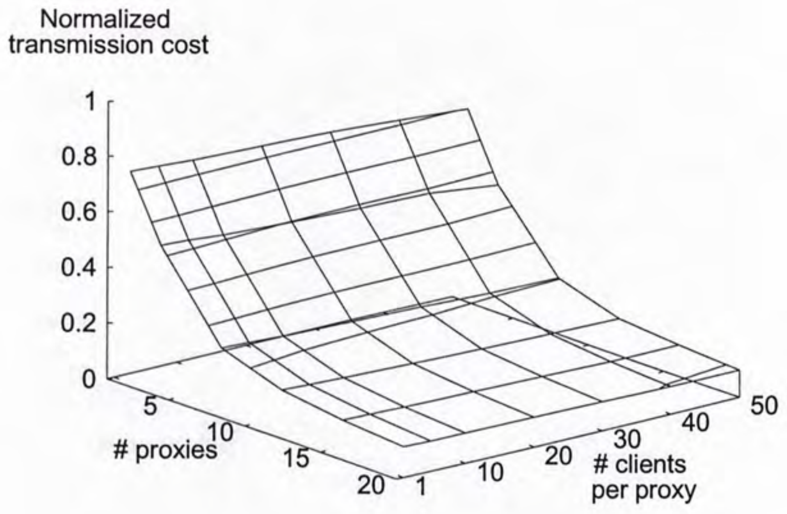


Figure 3.11: Transmission cost with different numbers of proxies and clients.

alently, the half size of one video, which can be easily accommodated by personal computers. With an increase of the number of proxies, we have observed a even more noticeable cost reduction, particularly when the number is changed from 1 to 5. This again confirms that proxy cooperation is worth considerations.

The control overhead is also an important concern toward realizing COPACC. We define the overhead of COPACC as the traffic volume of control messages (election, allocation, lookup, and retrieval, etc.) over the total traffic volume, which obviously depends on the scale and streaming rate of the system. In Fig. 3.12, we show the overhead with different number of proxies and streaming rates. The number of clients per proxy is set to 50. It can be seen that the overhead is reasonably low, which is less than 1% of the total traffic even with 20 proxies. In addition, the overhead decreases with higher streaming rates. This is mainly because the messages are quite short as compared to video segments, and most messages are locally exchanged.

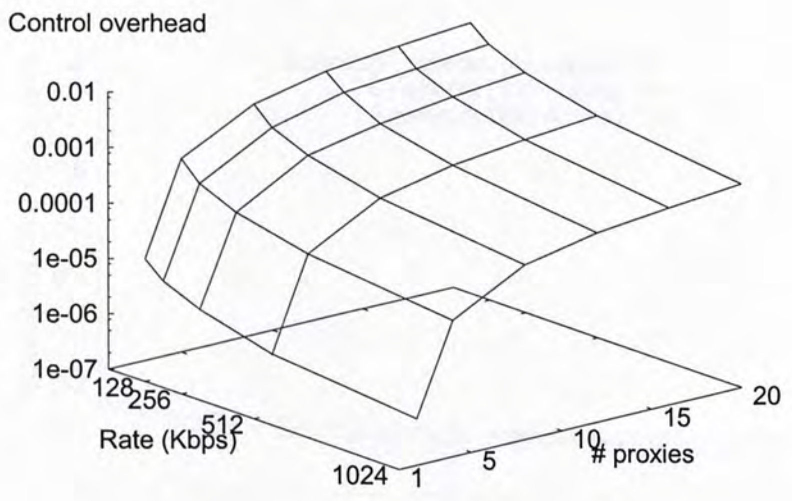


Figure 3.12: Control overhead with different number of proxies and streaming rates.

### 3.4.4 Sensitivity to Network Topologies

So far, we focus on regular network topologies with identical transmission costs between proxies. We have also investigated the performance of our system under various synthetic and real network topologies. Fig 3.13 shows the costs under three representative topologies: the 44-node SprintLink network and the 100- and 200-node Transit-Stub (TS) networks. The SprintLink network, representing the topology of a typical backbone network in north America, is obtained from the Rocketfuel project at the University of Washington [43]. The TS network is synthesized by the GT-ITM topology generator [53], which attempts to reproduce the hierarchical structure of the Internet by composing interconnected transit and stub domains. For both topologies, we randomly place the given number of proxies to the network nodes, and set the link cost inversely proportional to the bandwidth of each link. A shortest-path routing is then used to determine the path between

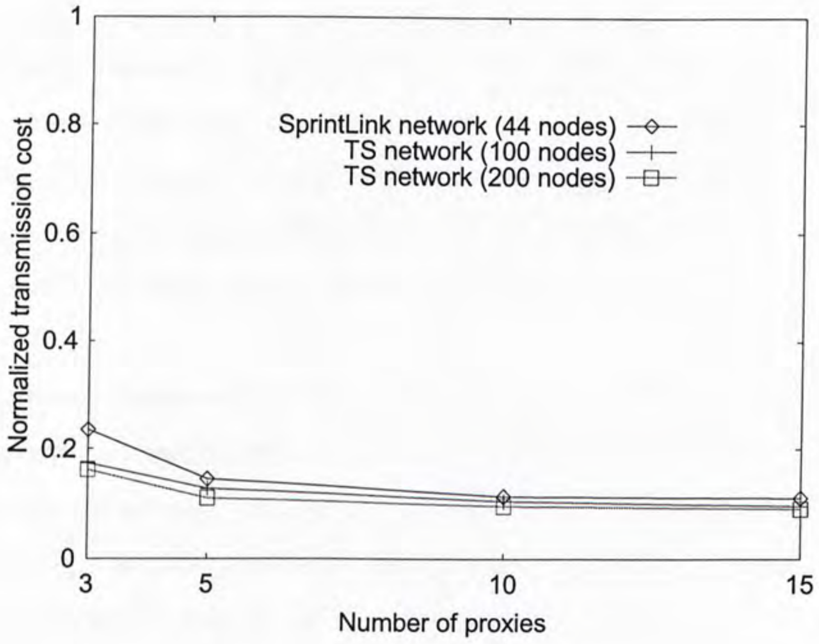


Figure 3.13: Transmission cost as a function of the number of proxies under real and synthetic network topologies.

proxies, and the cost of a path is the sum of costs across all the link of this path. The server is connected to these proxies through a remote link: in SprintLink network, it is assumed to be in Asia, and in TS network, we manually set the unit transmission cost to 5 times the average cost between proxies.

It can be seen that, under all the three network topologies, the transmission costs of COPACC are pretty low and generally decrease with an increase of the number of proxies. The performance under the TS topology is slightly better, suggesting that COPACC works well with a hierarchical network structure, where local transmission cost is much lower than remote transmission cost. It is worth noting that SprintLink network also follows a hierarchical structure, but many low-level nodes are abstracted into a single nodes. Moreover, the proxies in our evaluation are randomly placed. We thus expect a even better performance when the proxies are strategically placed and cooperated with each other in closer distances.

Overall, the evaluations demonstrate that COPACC achieves remarkably lower transmission cost as compared to a pure proxy-based caching with limited storage space. On the other hand, it is much more robust than a pure peer-to-peer communication system in the presence of node failures. Meanwhile, its computation and control overheads are both kept in low levels.

However, the merits of the COPACC rely on the active participation from the proxies and the clients. Thus, an incentive mechanism is essential to encourage the network entities to cooperate. Our Incentive-based COPACC achieves this by incorporating with a revenue-rewarding scheme to credit the proxies who contribute resource in the system.

□ **End of chapter.**

# Chapter 4

# Revenue-Rewarding Mechanism

In this chapter, we present an incentive mechanism for COPACC to encourage the proxies to participate. A revenue-rewarding scheme is proposed to reward part of the aggregated transmission cost saved to the contributing proxies. We start by giving an overview of the considered proxy caching system derived from COPACC. We model the interaction between the proxies under the revenue-rewarding scheme as a resource allocation game, and analyze the cache space contributed in a non-cooperative environment. We further suggest two cooperative games that achieve different system-wide properties. The performance of the three resource allocation games have been evaluated, and the results demonstrate that the revenue-rewarding scheme provides a strong incentive for different entities to cooperate in the network.

## 4.1 System Model

### 4.1.1 System Overview

We consider a cooperative proxy caching system for multimedia streaming. The architecture of this caching system is shown in Fig. 4.1. 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 Chapter 3, 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 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

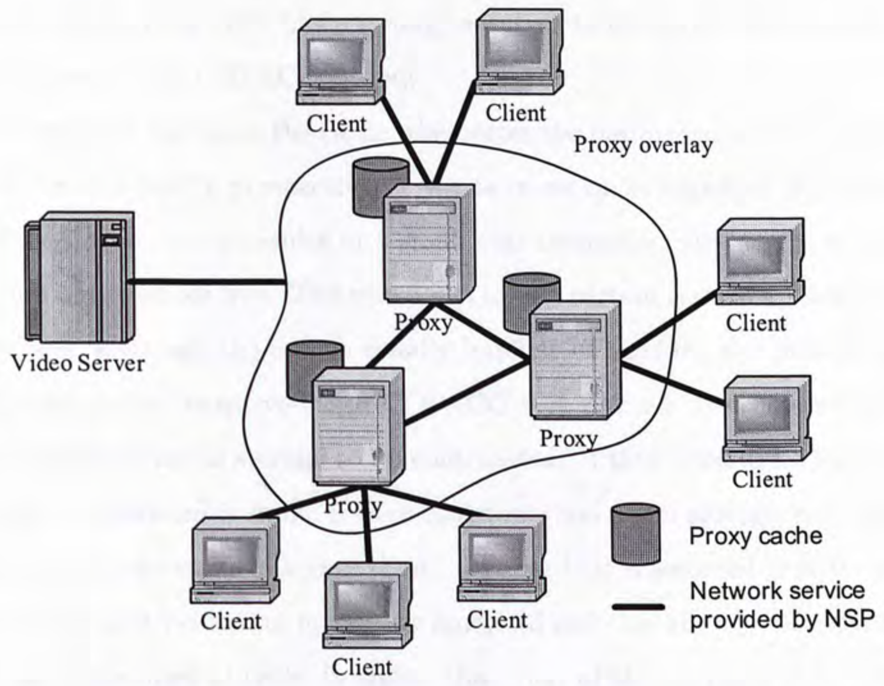


Figure 4.1: The architecture of the cooperative proxy caching system.

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 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

| **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 4.1: A summary of the notations.

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.

### 4.1.2 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 this chapter are summarized in Table 4.1.

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. $0 \leq s_i \leq \hat{S}_i$. 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 chapter, we consider the cost function to be strictly increasing with convex shape. The cost function for proxy $i$ is defined as follows:

$$C_i(s_i) = \begin{cases} A_i e^{\theta_i(s_i - b_i)}, & 0 < s_i \leq \hat{S}_i \\ 0, & s_i = 0. \end{cases} \tag{4.1}$$

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 $\theta_i$ determines the increasing rate of the cost. For example, a cost function with large value of $A_i$ and $\theta_i$ have a high cost. When $\theta_i$ 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 $A_i$, $\theta_i$ 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

$$R(q) = \frac{A'}{\theta'}[1 - e^{-\theta'(q-b')}], \quad q > 0. \tag{4.2}$$

In practice, $A'$ and $\theta'$ are greater than zero, and $A'/\theta'$ should be a finite number, as it represents the revenue obtained when there is infinite amount of cache. It is less likely that $\theta'$ approaches to zero, in which the revenue also approaches to zero.

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 follows:

$$P(q) = \frac{A'}{\theta' q}[1 - e^{-\theta'(q-b')}], \quad q > 0. \tag{4.3}$$

$$P(q) = A' e^{-\theta'(q-b')}, \quad q > 0 \tag{4.4}$$

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.

## 4.2 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:

$$U_i(s_i) = s_i P(q) - C_i(s_i). \tag{4.5}$$

The resource allocation game is a repeated synchronous 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.

### 4.2.1 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 $[0, \hat{S}_i]$:

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = s_i P(q) - C_i(s_i). \tag{4.6}$$

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 follows:

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = s_i P(s_i + s_{-i}) - C_i(s_i). \tag{4.7}$$

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

$$\max_{0 \le s_i \le \hat{S}_i} U_i(s_i) = \begin{cases} s_i A' e^{-\theta'(s_i + s_{-i} - b')} - A_i e^{\theta_i (s_i - b_i)}, & 0 < s_i \le \hat{S}_i \\ 0, & s_i = 0. \end{cases} \tag{4.8}$$

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*.

### 4.2.2 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.

$$E(q) = R(q) - qP(q). \tag{4.9}$$

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 (4.9), 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[5] 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:

$$\max_{0 \leq s_i \leq \hat{S}_i} U_i(s_i) = s_i P(q) - C_i(s_i) = s_i p - C_i(s_i). \tag{4.10}$$

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. $0 \leq s_i \leq \hat{S}_i$. 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

$$\max_{0 \leq s_i \leq \hat{S}_i} U_i(s_i) = U_i({s_i}^*). \tag{4.11}$$

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

$$U_i'(s_i) = p - C_i'(s_i) = 0 \tag{4.12}$$

$$C_i'(s_i) = A_i \theta_i e^{\theta_i (s_i - b_i)} = p \tag{4.13}$$

$$s_i = \frac{ln(p/A_i\theta_i)}{\theta_i} + b_i. \tag{4.14}$$

By solving $s_i$ in Equation (4.13), one can obtain the solution of $s_i^*$.

$$s_i^* = \begin{cases} 0, & s_i \le 0 \\ s_i, & 0 < s_i \le \hat{S}_i \\ \hat{S}_i, & s_i > \hat{S}_i. \end{cases} \tag{4.15}$$

Obviously, there is only one value of $s_i^*$ that can satisfy the objective function in Equation (4.11). 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 4.2.2.1.** *The profit maximizing game admits one and only one Nash equilibrium*

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

$$U_i''(s_i) = -C_i''(s_i). \tag{4.16}$$

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 (4.13) 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

$$q^* = \sum_{i=1}^{H} s_i^*. \tag{4.17}$$

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

$$\max_{p \geq 0} E(q^*) = R(q^*) - q^* p \tag{4.18}$$

Since the total amount of cache space $q^*$ is solely depended on the value of price $p$ through Equation (4.15) and (4.17), 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 (4.19).

$$\max_{p \geq 0} E_p(p) = R(\sum_{i=1}^{H}(\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i)) - p \cdot \sum_{i=1}^{H}(\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i) \tag{4.19}$$

The derivative of $q^*$ and $E_p(p)$ with respect to $p$ are shown below. The optimal price, $p^*$, can be obtained by solving the first-order condition in Equation (4.23).

$$\frac{dq^*}{dp} = \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} \tag{4.20}$$

$$E_p'(p) = A'e^{-\theta'(q^*-b')} \cdot \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} - q^* - p \cdot \frac{1}{p}\sum_{i=1}^{H}\frac{1}{\theta_i} \tag{4.21}$$

$$= \frac{A'}{p}(\sum_{i=1}^{H}\frac{1}{\theta_i})e^{-\theta'(q^*-b')} - q^* - \sum_{i=1}^{H}\frac{1}{\theta_i} \tag{4.22}$$

$$E_p'(p) = 0. \tag{4.23}$$

Although it is hard to find the close-form solution of the optimal price $p^*$ for Equation (4.23), 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 (4.15). 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 $E_p(p)$, 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 (4.25)-(4.30).

$$L = E_p(p) - \sum_{i=1}^{H} \mu_i^l s_i + \sum_{i=1}^{H} \mu_i^u (s_i - \hat{S}_i) \tag{4.24}$$

$$\frac{\partial L}{\partial p} = \frac{\partial E_p(p)}{\partial p} - \sum_{i=1}^{H} \mu_i^l \frac{\partial s_i}{\partial p} + \sum_{i=1}^{H} \mu_i^u (\frac{\partial s_i}{\partial p} - \hat{S}_i) = 0 \tag{4.25}$$

$$\mu_i^l \geq 0, \quad i = 1, ..., H \tag{4.26}$$

$$\mu_i^u \geq 0, \quad i = 1, ..., H \tag{4.27}$$

$$\mu_i^l s_i = 0, \quad i = 1, ..., H \tag{4.28}$$

$$\mu_i^u (s_i - \hat{S}_i) = 0, \quad i = 1, ..., H \tag{4.29}$$

$$0 \leq s_i \leq \hat{S}_i, \quad i = 1, ..., H. \tag{4.30}$$

We now present an algorithmic approach to find the optimal value of $p$, which is derived directly from the KKT-condition. Fig. 4.2 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 (4.28) and (4.29) hold only if the $\mu_i^l$ and $\mu_i^u$ are zero. The optimization problem is now similar to the unconstrained problem in Equation (4.19), 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 $\mu_i^l$ or $\mu_i^u$ 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 (4.28) or (4.29). We can identify the active constraints of $s_i$ from the result obtained in Equation (4.23). 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.

**Profit Maximizing Algorithm:**

```
1:  declare P = { 1, 2, ... , $H$}; // indexes of proxy that it's $s_i$
                                       has not been determined yet
2:  declare $P^l = \{\}$;   // indexes of proxy that it's $s_i$ is zero
3:  declare $P^u = \{\}$;   // indexes of proxy that it's $s_i$ is $\hat{S}_i$
4:  for $i := 1$ to $H$
5:    $s_i = 0$;
6:  end for
7:  while (true) do
8:    $q = \sum_{i \in P} s_i + \sum_{i \in P^u} \hat{S}_i = \sum_{i \in P}[\frac{ln(p/A_i\theta_i)}{\theta_i} + b_i] + \sum_{i \in P^u} \hat{S}_i$;
9:    Solve the optimal price $p$ that maximize $E_p(p) = R(q) - p \cdot q$
      (or find $p$ s.t. $E_p'(p) = 0$);
10:     for $i := 1$ to $H$
11:       $s_i = ln(p/A_i\theta_i)/\theta_i + b_i$;
12:     end for
13:     if $0 \leq s_i \leq \hat{S}_i \; \forall i \in P$ then
14:       break;   // end the while loop
15:     end if
16:     declare $P^t = \{\}$;   // a temporary set
17:     for $i := 1$ to $H$
18:       if $s_i \leq 0$ then
19:         $P^t = P^t \cup \{i\}$;
20:       end if
21:     end for
22:     if $(P^t - P^l) \neq \emptyset$ then
23:       $P^l = P^t$;
24:       $P = P - P^t$;
25:       continue;   // next iteration of the while loop
26:     end if
27:     for each $i \in P$
28:       if $s_i \geq \hat{S}_i$ then
29:         $P^u = P^u \cup \{i\}$;
30:         $P = P - \{i\}$;
31:       end if
32:     end for
33:   end while
34:   return $p$;   // $p$ is the optimal price
```

Figure 4.2: Profit Maximizing Algorithm for the NSP in the Profit Maximizing Game.

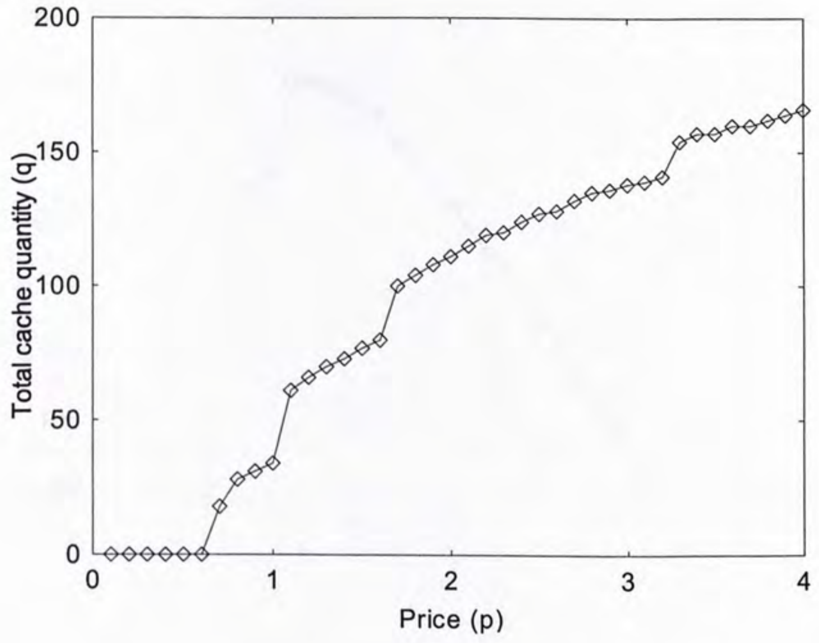


Figure 4.3: A sample plot of total cache space $q$ versus price $p$.

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. 4.3 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. 4.4 plots a sample relationship between the net profit $E_p(p)$ and the price. We observe that the value of $E_p(p)$ in Equation (4.19) 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 (4.10), 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

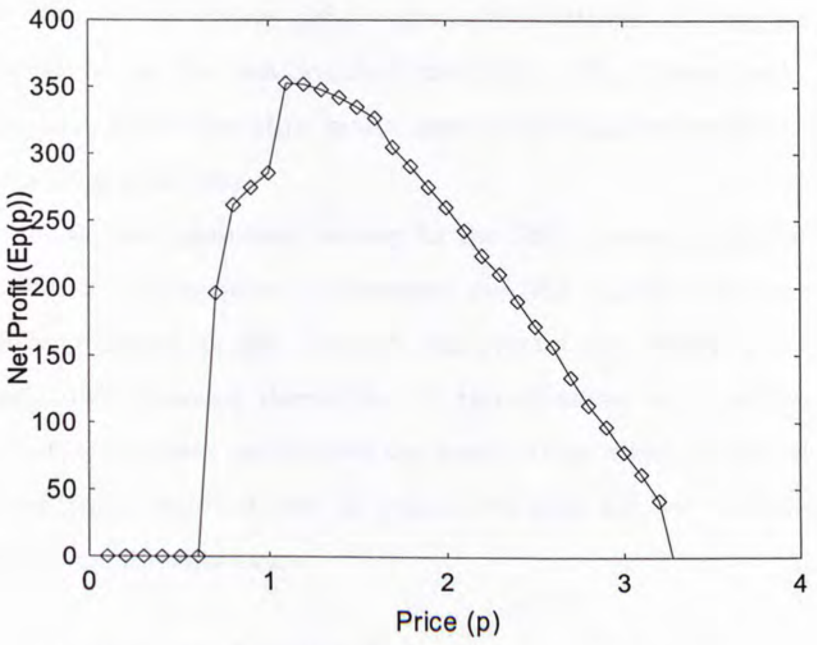


Figure 4.4: A sample plot of net profit $E_p(p)$ versus price $p$.

it agrees to contribute. Based on the total cache space $q$ supplied by the proxies, the NSP decides the best pricing 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 $E_p(\dot{p})$ 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 Estab-*

*lishing Protocol* to achieve global optimization with fast converging speed. We suggest to use Pattern Search Method [33] 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.

### 4.2.3 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 subsection, 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 (4.5). We define the social utility as the individual utility summed over all proxies, which is

$$SU(s_1, s_2, ..., s_H) = \sum_{i=1}^{H} U_i(s_i). \tag{4.31}$$

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

$$\max_{0 \leq s_i \leq \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0 \leq s_i \leq \hat{S}_i} \sum_{i=1}^{H} [s_i P(q) - C_i(s_i)]. \tag{4.32}$$

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 (4.32) can be simplified as:

$$\max_{0 \le s_i \le \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0 \le s_i \le \hat{S}_i} \left[ q \cdot P(q) - \sum_{i=1}^{H} C_i(s_i) \right]. \tag{4.33}$$

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 (4.32) can be rewritten as below:

$$\max_{0 \le s_i \le \hat{S}_i} \sum_{i=1}^{H} U_i(s_i) = \max_{0 \le q \le \hat{Q}} \left[ q \cdot P(q) - \min_{0 \le s_i \le \hat{S}_i} \sum_{i=1}^{H} C_i(s_i) \right] \tag{4.34}$$

$$s.t. \qquad q = \sum_{i=1}^{H} s_i \tag{4.35}$$

$$\hat{Q} = \sum_{i=1}^{H} \hat{S}_i. \tag{4.36}$$

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.

### 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:

$$\textbf{MinCost:} \quad MinCost(q) = \min \sum_{i=1}^{H} C_i(s_i) \tag{4.37}$$

$$s.t. \quad 0 \le s_i \le \hat{S}_i, \qquad i = 1, ..., H \tag{4.38}$$

$$\sum_{i=1}^{H} s_i = q. \tag{4.39}$$

The formulation like Equation (4.37) 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 $1 \leq i \leq H$ and $0 \leq j \leq q$.

$$B(i,j) = \begin{cases} C_1(j), & 0 \leq j \leq \hat{S}_1 \\ \infty, & i = 1,\ \hat{S}_1 < j \leq q \\ \min\limits_{0 \leq k \leq j, k \leq \hat{S}_i} B(i-1, j-k) + C_i(k), & 2 \leq i \leq H,\ 0 \leq j \leq q. \end{cases} \tag{4.40}$$

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 $M = \max_{1 \leq i \leq H} \hat{S}_i$.

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 (4.37), it is proven as a constrained convex optimization problem.

**Theorem 4.2.3.1.** *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,

$$\nabla^2[\sum_{i=1}^{H} C_i(s_i)] = \begin{bmatrix} C_1''(s_1) & 0 & 0 & 0 \\ 0 & C_2''(s_2) & 0 & 0 \\ 0 & 0 & C_3''(s_3) & 0 \\ 0 & 0 & 0 & ... \end{bmatrix} \tag{4.41}$$

or

$$\nabla^2[\sum_{i=1}^{H} C_i(s_i)]_{ij} = \begin{cases} 0, & i \neq j \\ C_i''(s_i), & i = j. \end{cases} \tag{4.42}$$

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:

$$L(s_i) = \sum_{i=1}^{H} C_i(s_i) - \lambda(\sum_{i=1}^{H} s_i - q) - \sum_{i=1}^{H} \mu_i^l s_i + \sum_{i=1}^{H} \mu_i^u (s_i - \hat{S}_i) \tag{4.43}$$

$$\frac{\partial L}{\partial s_i} = C_i'(s_i) - \lambda - \mu_i^l + \mu_i^u = 0, \quad i = 1, ..., H \tag{4.44}$$

$$\mu_i^l \geq 0, \quad i = 1, ..., H \tag{4.45}$$

$$\mu_i^u \geq 0, \quad i = 1, ..., H \tag{4.46}$$

$$\mu_i^l s_i = 0, \quad i = 1, ..., H \tag{4.47}$$

$$\mu_i^u (s_i - \hat{S}_i) = 0, \quad i = 1, ..., H \tag{4.48}$$

$$0 \leq s_i \leq \hat{S}_i, \quad i = 1, ..., H \tag{4.49}$$

$$\sum_{i=1}^{H} s_i = q. \tag{4.50}$$

The MinCost is a convex optimization problem, meaning that there exists a unique solution $\{s_i\}_{i=1}^H$ satisfying the Equation (4.44)-(4.50). 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 (4.37), 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 $\mu_i^l$ and $\mu_i^u$ omitted. Note that the plus or minus sign of the multiplier term does not affect the solution.

$$L(s_i) = \sum_{i=1}^{H} C_i(s_i) - \lambda(\sum_{i=1}^{H} s_i - q) \tag{4.51}$$

$$\frac{\partial L}{\partial s_i} = C_i'(s_i) - \lambda = 0, \quad i = 1, ..., H \tag{4.52}$$

$$\sum_{i=1}^{H} s_i = q. \tag{4.53}$$

We instantiate the cost function according to the COPACC system. Given Equation (4.52), we can derive $s_i$ in terms of $\lambda$.

$$C_i'(s_i) = A_i\theta_i e^{\theta_i(s_i - b_i)} = \lambda \tag{4.54}$$

$$s_i = \frac{ln(\lambda/A_i\theta_i)}{\theta_i} + b_i. \tag{4.55}$$

The variable $\lambda$ 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 (4.14), but in here we have one more condition about the total cache quantity (in Equation (4.53)) to hold. By substituting $s_i$ to Equation (4.53), we can solve the value of $\lambda$, and thus, the values of all $s_i$.

$$\sum_{i=1}^{H} \left[ \frac{ln(\lambda/A_i\theta_i)}{\theta_i} + b_i \right] = q \tag{4.56}$$

$$\lambda = e^{\frac{q + \sum_{i=1}^{H} [ln(A_i\theta_i)/\theta_i - b_i]}{\sum_{i=1}^{H} 1/\theta_i}}. \tag{4.57}$$

If we have all the parameters about the individual cost function of each proxy, we can find the optimal $\lambda$ as well as the $s_i$ directly. However, in the distributed approach, we must rely on iteratively refining the value of $\lambda$ 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.

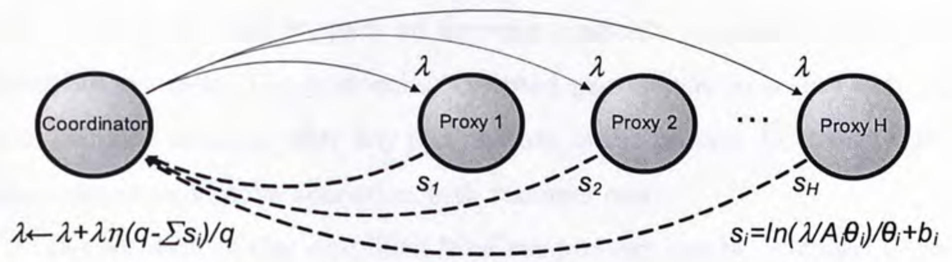


Figure 4.5: Mechanism of the cost minimization protocol.

The coordinator first make an initial guess of the shadow price $\lambda$, and notifies the proxies. Each proxy reacts to the shadow price with a $s_i$ obtained by Equation (4.55), 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. 4.5 shows the mechanism of the cost minimization protocol. We adopt to a simple updating rule, which increase/decrease the value of $\lambda$ in proportional to the difference between the desirable cache space $q$ and the total contributed cache from the proxies. The updating rule of $\lambda$ is

$$\lambda \leftarrow \lambda + \lambda\eta(\frac{q - \sum_{i=1}^{H} s_i}{q}). \tag{4.58}$$

The learning rate, $\eta$, is a factor that controls the converging speed and the accuracy of the protocol. A large value of $\eta$ 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 $\lambda$ 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 (4.47) in the KKT-condition, either $\mu_i^l$ equals zero or $s_i$ equals zero. Similarly in Equation (4.48), either $\mu_i^u$ equals zero or $s_i$ equals $\hat{S}_i$. To solve the set of linear equations, we have to examine whether $\mu_i^l$ and $\mu_i^u$ are equal to zero. Assume both $\mu_i^l$ and $\mu_i^u$ 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 (4.49). In that case, depending on the value of $s_i$, one of the $\mu_i^l$ and $\mu_i^u$ 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 $\mu_i^l$ 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. 4.6.

```
Cost Minimization:
1:  declare P = { 1, 2, ... , $H$};  // indexes of proxy that it's $s_i$
                                        has not been determined yet
2:  declare $P^l = \{\}$;   // indexes of proxy that it's $s_i$ is zero
3:  declare $P^u = \{\}$;   // indexes of proxy that it's $s_i$ is $\hat{S}_i$
4:  while (true) do
5:    the new cache requirement $q' = q - \sum_{i \in P^u} \hat{S}_i$;
6:    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 $i \in P$;
7:    if $\forall i \in P$, $0 \leq s_i \leq \hat{S}_i$ then
8:      break;   // end the while loop
9:    end if
10:   declare $P^t = \{\}$;   // a temporary set
11:   for each $i \in P$
12:     if $s_i \leq 0$ then
13:       $P^t = P^t \cup \{i\}$;
14:     end if
15:   end for
16:   if $P^t \neq \emptyset$ then
17:     $P^l = P^l \cup P^t$;
18:     $P = P - P^t$;
19:     continue;   // next iteration of the while loop
20:   end if
21:   for each $i \in P$
22:     if $s_i \geq \hat{S}_i$ then
23:       $P^u = P^u \cup \{i\}$;
24:       $P = P - \{i\}$;
25:     end if
26:   end for
27: end while
```

Figure 4.6: Cost Maximization Protocol for the proxy coordinator.

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.

### 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

$$\textbf{OptQ:} \quad \max_{0 \leq q \leq \hat{Q}} [q \cdot P(q) - M(q)] \tag{4.59}$$

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

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. 4.7). In fact, the objective function may contain multiple maxima, depending on the cost functions and revenue function used in the system. Fig. 4.8 plots the revenue, minimum cost and social utility with respect to cache quantity in the COPACC 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.

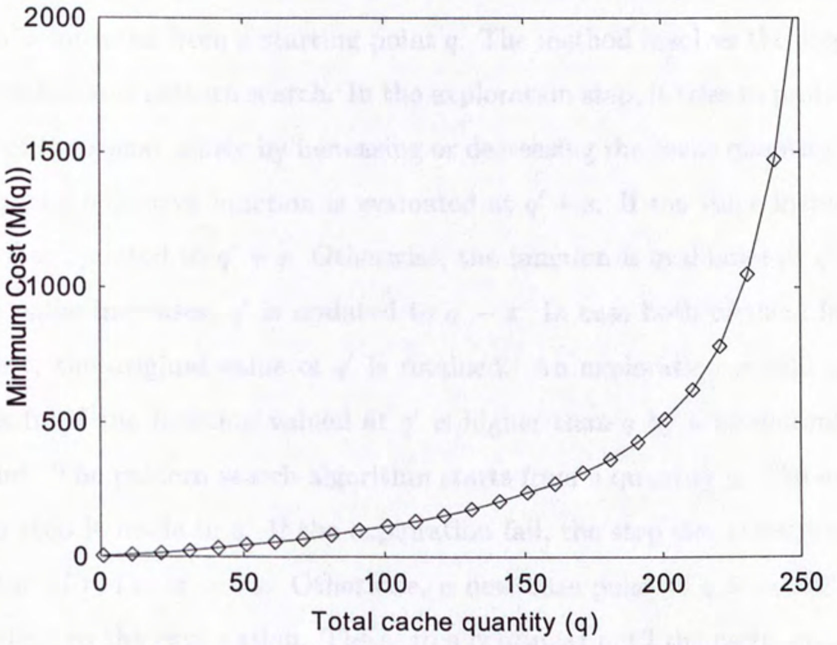


Figure 4.7: A sample plot of minimum cost versus cache quantity $q$.

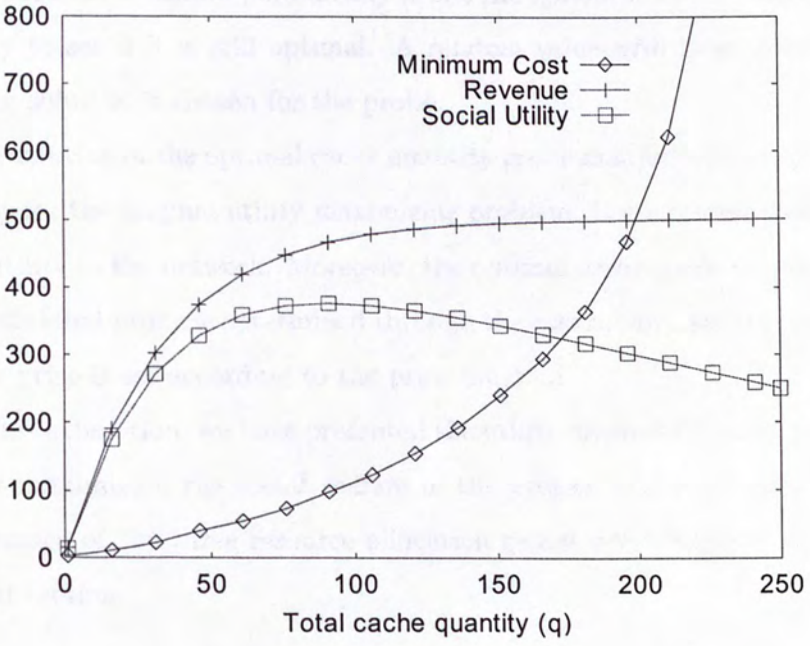


Figure 4.8: A sample plot of social utility versus cache quantity $q$.

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. The solution obtained may not be global maximum. To ensure that the solution does not trap in the local maxima at steady state, the proxy coordinator should periodically probe the system with different cache quantity to see if it is still optimal. A random value with large difference from the solution is chosen for the probe.

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.

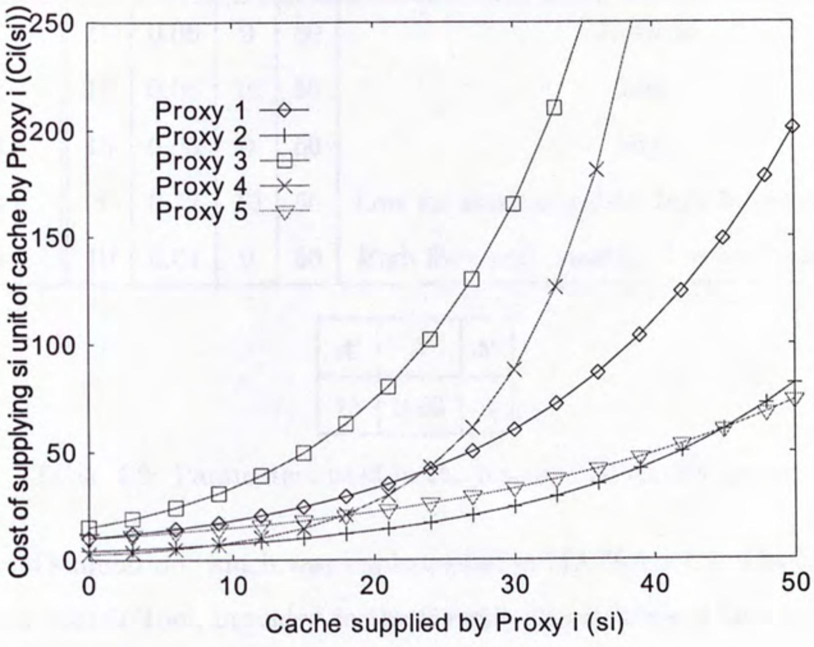
In this subsection, we have presented the utility maximizing game, which aims at maximizing the social welfare of the proxies in the network. The performance of the three resource allocation games are being evaluated in the next section.

## 4.3 Performance Evaluation

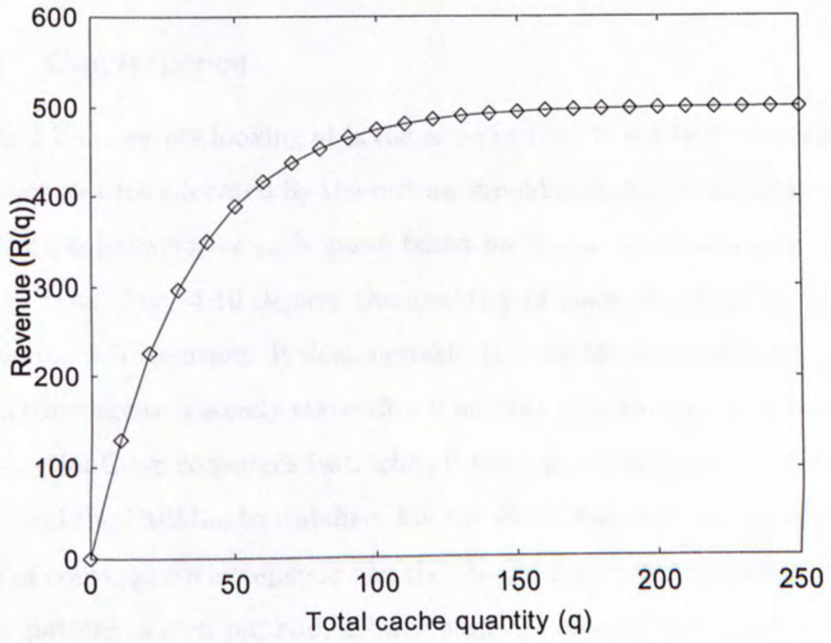
The main focus of this section is to evaluate the effectiveness of the proposed revenue-rewarding scheme in encouraging the participation of the NSP and the proxies. We show that the scheme can provide a strong incentive for different entities to join the system. We have examined the use of revenue-rewarding 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 4.2, and the corresponding functions are plotted in Fig. 4.9. 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 math-



(a) Cost functions of the proxies.



(b) Revenue function of the system.

Figure 4.9: Cost and revenue functions being used in the evaluation.

| Proxy $i$ | $A_i$ | $\theta_i$ | $b_i$ | $\hat{S}_i$ | 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 |

| $A'$ | $\theta'$ | $b'$ |
|---|---|---|
| 15 | 0.03 | 0 |

Table 4.2: Parameters used in the resource allocation game.

ematical 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.

### 4.3.1 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. 4.10 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 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.

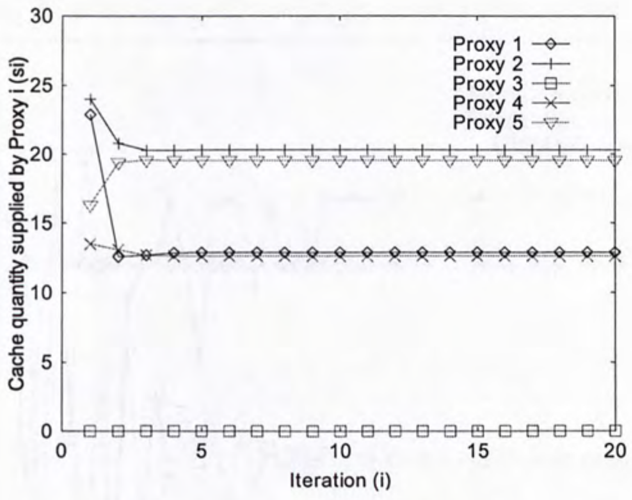
### 4.3.2 Participation Incentive

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.

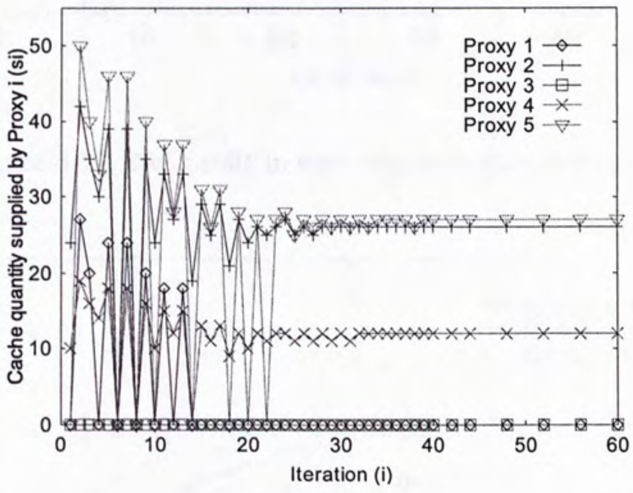
#### Net Profit

Fig. 4.11 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.

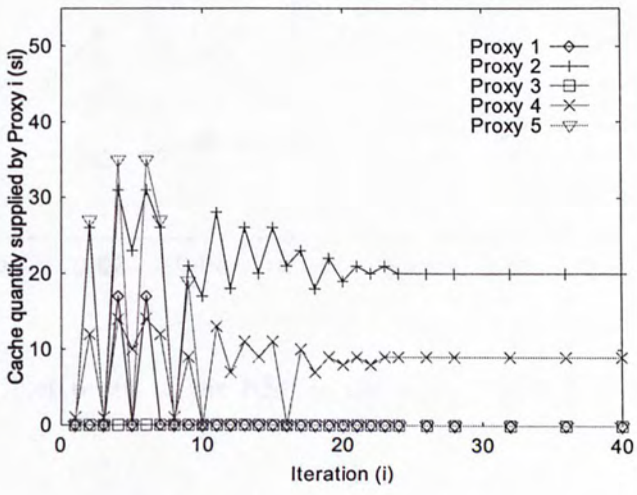
We also evaluated the performance of the three games under systems having different revenue function. All the revenue functions had the same ratio of $A'/\theta'$, but the value of $\theta'$ varied from 0.01 to 0.08. Remember that the larger the $\theta'$, the higher the revenue is for the same quantity of cache. The net profit of the NSP in the systems with different revenue function is plotted in Fig. 4.12. 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

Figure 4.10: Quantity of cache supplied by each proxy.

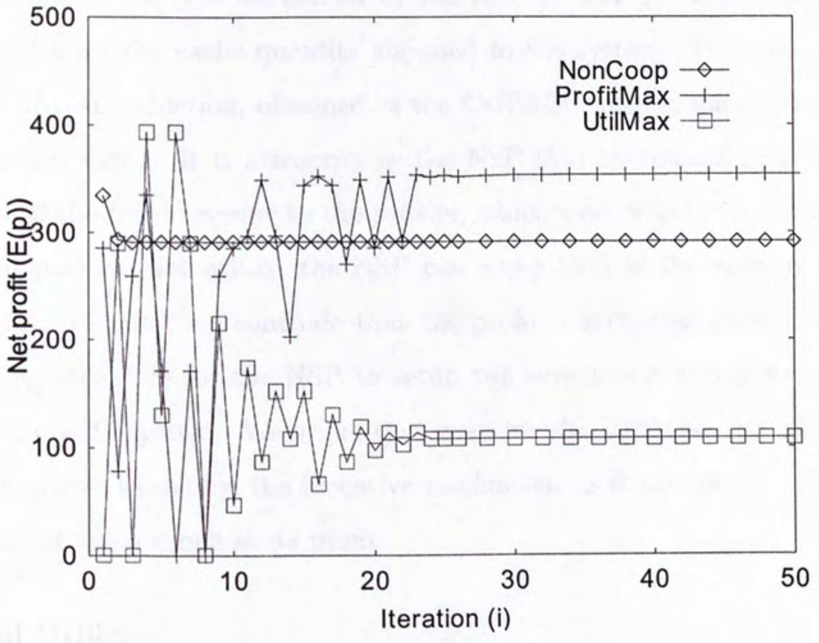


Figure 4.11: Net profit in each resource allocation game.

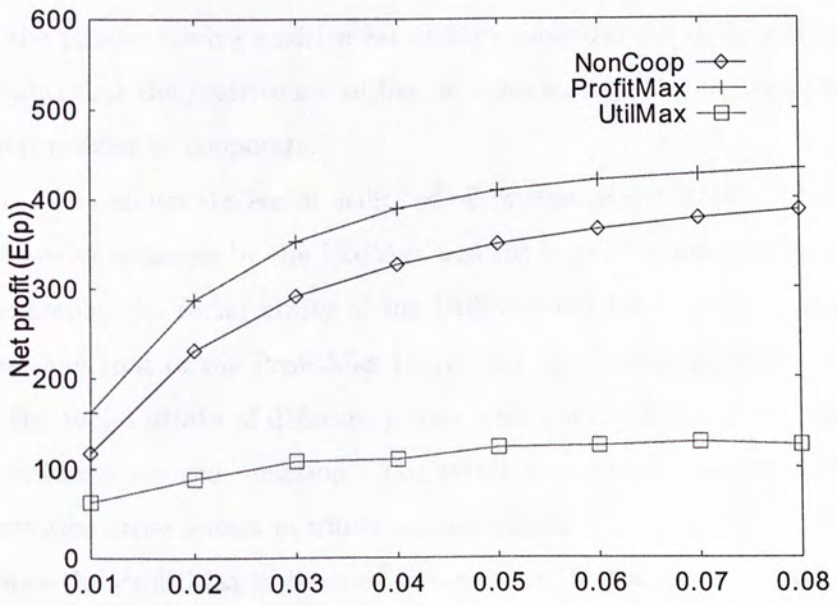


Figure 4.12: Net profit of the NSP in the system with $\theta'$ varied from 0.01 to 0.08.

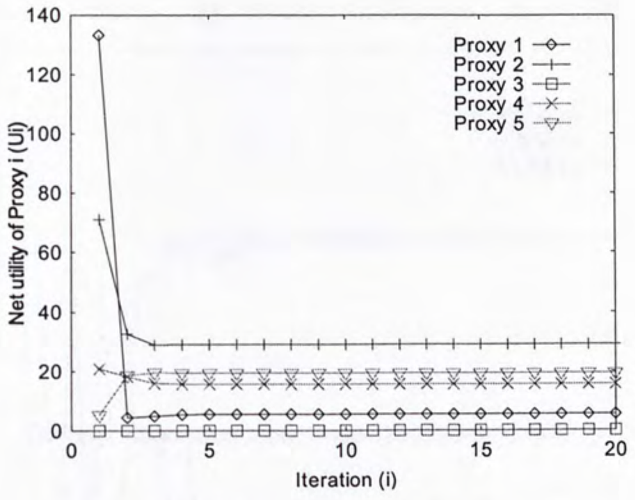
Note that the revenue earned by the NSP in each game was different, depending on the cache quantity supplied to the system. The revenue, in terms of cost reduction, obtained in the COPACC system under the ProfitMax was 426.7. It is attractive to the NSP that by rewarding 17% (i.e. 73.60) of the total 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-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.

### Social Utility

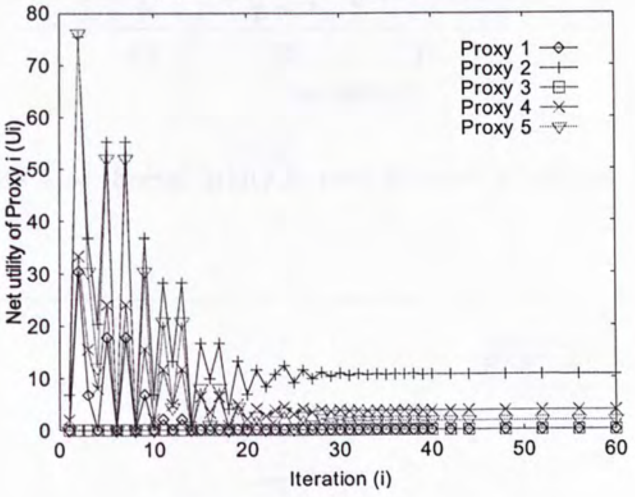
In this subsection, we evaluate the social utility in different games. Fig. 4.13 demonstrates the individual utility of the proxies under different games. Only the proxies having positive net utility supplied cache to the system. It illustrates that the positive net utility provides an initiative incentive to the rational proxies to cooperate.

Fig. 4.14 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. 4.15, 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.

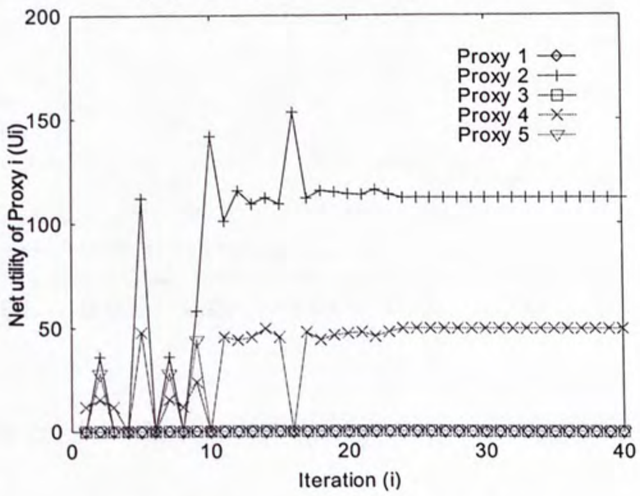
Since the ProfitMax is designed to maximize the net profit by trading off the social utility, the utility achieved in the ProfitMax is the lowest. Note that the social utility of the ProfitMax and the NonCoop decreased with an increase of $\theta'$. It shows that more utility is traded for the net profit when



(a) Non-cooperative Game



(b) Profit Maximizing Game



(c) Utility Maximizing Game

Figure 4.13: Net utility of each proxy.

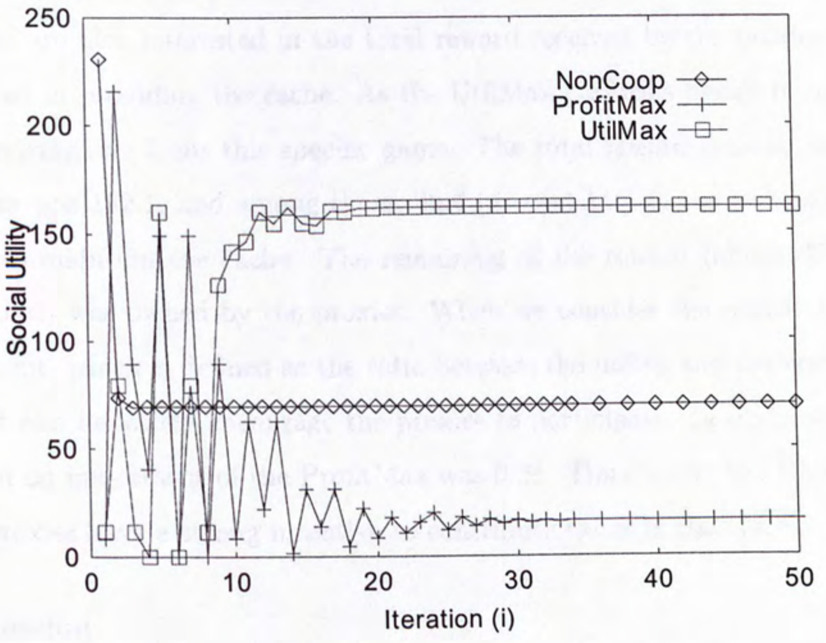


Figure 4.14: Social utility in each resource allocation game.

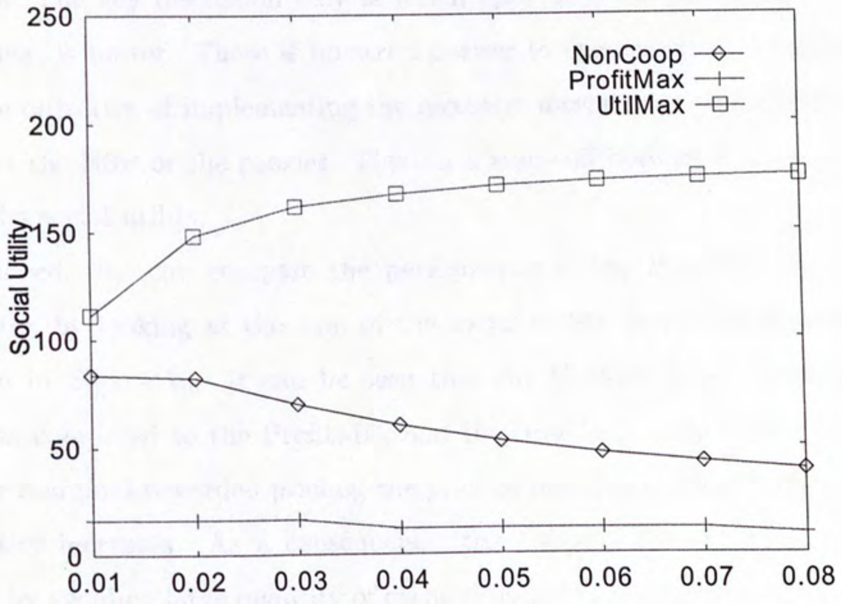


Figure 4.15: Social utility in the system with $\theta'$ varied from 0.01 to 0.08.

the $\theta'$ 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.

### 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. 4.16. 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

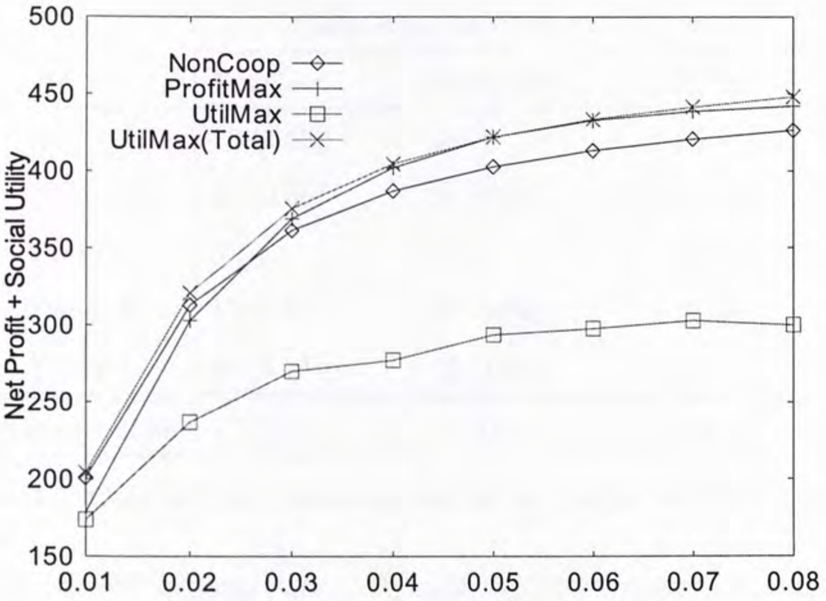


Figure 4.16: Sum of the social utility and the net profit in each resource allocation game with $\theta'$ varied from 0.01 to 0.08.

total-rewarded pricing, in which the net profit of the NSP is always zero. The UtilMax(Total) showed in Fig. 4.16 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 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

$$\max_{0 \le s_i \le \hat{S}_i} \left[ E(q) + \sum_{i=1}^{H} U_i(s_i) \right] = \max_{0 \le s_i \le \hat{S}_i} \left[ R(q) - \sum_{i=1}^{H} C_i(s_i) \right]. \tag{4.60}$$

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-

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

Table 4.3: Cost per unit cache supplied by the proxies in different game.

coordinated P2P-like application. By applying the revenue-rewarding scheme, the network entities are stimulated to participate in the system.

### 4.3.3 Cost effectiveness

It can be seen in Fig. 4.10 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 4.3 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 4.3. 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.

□ **End of chapter.**

# Chapter 5

# Conclusion

In this thesis, we have introduced a cooperative and incentive-based proxy-and-client caching system for on-demand media streaming. Two major components: Cooperative proxy-and-client caching and Revenue-rewarding mechanism, have been discussed.

In summary, COPACC is a novel cooperative proxy-and-client caching system that combines the best features of proxy caching and peer-to-peer communications. It leverages the client-side caching to amplify the aggregated cache space and relies on dedicated proxies to effectively coordinate the communications. We have developed an efficient cache allocation algorithm for distributing video segments among the proxies and clients. A comprehensive suite of protocols are presented to facilitate the interactions among different network entities. It also enables smart and cost-effective cache indexing, searching, verifying operations in this hybrid caching system. However, COPACC does not address the incentive issue. That is, what motivates each proxy to provide cache space and how much cache space should be allocated. We have extended COPACC by suggesting 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. 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 cooperative and incentive-based proxy-and-client caching system under various network and end-system configurations. Our key findings can be summarized as follows:

1. With an amplified total cache spaces, cooperative proxy-and-client caching significantly reduces the transmission cost for on-demand media streaming.

2. With the assistance from dedicated proxies, it is much more robust than a pure peer-to-peer system, even though the proxies may contribute only a small fraction of the total cache space.

3. COPACC scales well in larger network, and the cost generally reduces when more proxies and clients cooperate with each other.

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

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

6. The two cooperative games yield a cost-effective resource allocation from the proxies.

In the future, we can perform more experiments to compare the performance of COPACC to other P2P streaming systems, such as CoolStreaming

[54] and Loopback [34]. The reliability of COPACC can also be enhanced by considering replication of cache among proxies. Note the incentive mechanism is only applied to the proxies in the system, we can further extend the mechanism to encourage the clients to participate.

□ **End of chapter.**

# Appendix A

# NP-Hardness of the CAP problem

In this appendix, we prove the NP-hardness of the general optimal cache allocation problem (CAP). We show this by transforming the optimal resource allocation problem (RAP) to CAP in polynomial time. It is known that RAP is NP-hard and its decision version is NP-complete [31].

In RAP, there are $M$ kinds of resources to be allocated to $N$ activities, indexed from 1 through $N$, and the total available quantity of resource $j(\in [1 \dots M])$ is $N_j$. The objective is to minimize the cost in allocating the resources to activities, which can be formulated as:

$$\begin{aligned} &\textbf{RAP}: \min \textstyle\sum_{i=1}^{N} f_i(\sum_{j=1}^{M} a_{ij}x_{ij}), \\ &\text{s.t.} \textstyle\sum_{i=1}^{N} x_{ij} \le N_j, j = 1, 2, ..., m, \\ &\quad x_{ij} \in Z^+, \end{aligned}$$

where $x_{ij}$ is the quantity of resource $j$ allocated to activity $i$, $a_{ij}$ is the effectiveness for each unit of resource $j$ allocated to activity $i$, and $f_i()$ is a convex and non-increasing cost function for activity $i$ with given allocations.

Note that the resources and activities in RAP are analogous to the cache spaces and videos in CAP, respectively. Given an instance of RAP, we can create a CAP problem with the following settings: $s_j^p = N_j$ ,$s_{j,k}^c = 0$, and $p_j^i = x_{ij}$, $i \in [1...N], j \in [1...H], k \in [1...K_j]$. Since $Cost(\{p_j^i\}, \{q_{j,k}^i\})$ can be arbitrary function, we set it as $f_i(\sum_{j=1}^{M} a_{ij} p_j^i)$. We further set $V^i$ to $\sum_{j=1}^{M} N_j$, such that the constraint $\sum_{j=1}^{H} p_j^i + \sum_{j=1}^{H} \sum_{k=1}^{K_j} q_{j,k}^i \leq V^i$ in CAP is always satisfied. Given this transformation, it is obvious that an optimal solution to CAP, $p_j^i$, leads to an optimal solution to RAP: $x_{ij} = p_j^i, i \in [1...N], j \in [1..H]$. Since transformation is in polynomial time, it follows that problem CAP is NP-hard.

□

# Appendix B

# Optimality of the Greedy Algorithm

In this appendix, we prove the optimality of the proposed greedy algorithm for **PA** with $f_1^i = f_2^i = ... = f_H^i$.

We define the matrix of the unit transmission costs $\bar{W}^p(i,j)$ after executing step 1 through 2 as:

$$\begin{pmatrix} \bar{W}^p(1,1) & \bar{W}^p(1,2) & \dots & \dots & \bar{W}^p(1,H) \\ \bar{W}^p(2,1) & \bar{W}^p(2,2) & \dots & \dots & \bar{W}^p(2,H) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \bar{W}^p(i,j) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \bar{W}^p(N,1) & \bar{W}^p(N,2) & \dots & \dots & \bar{W}^p(N,H) \end{pmatrix}$$

where $\bar{W}^p(i,j) = \sum_{j'=1}^{H} \left[ w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p} \right] \lambda_{j'} f_{j'}^i$. Since $f_j^i = f_{j'}^i$ for all $j \neq j'$, we can drop subscript $j$ of $f_j^i$ and simplify the calculation of $\bar{W}^p(i,j)$ as $f^i \cdot \sum_{j'=1}^{H} \left[ w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p} \right] \lambda_{j'}$. We have the following two observations on $\bar{W}^p(i,j)$:

**Observation 1.** Given $i, \hat{i} \in [1 \dots N]$, $\bar{W}^p(i,j)/\bar{W}^p(\hat{i},j)$ is a constant for any $j \in [1 \dots H]$.

*Proof.*
$$\begin{aligned}
&\bar{W}^p(i,j)/\bar{W}^p(\hat{\imath},j)\\
&= f^i \cdot \textstyle\sum_{j'=1}^{H} \left[w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'} / f^{\hat{\imath}} \cdot \textstyle\sum_{j'=1}^{H} \left[w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'}\\
&= f^i / f^{\hat{\imath}}\\
&= \bar{W}^p(i,\hat{\jmath})/\bar{W}^p(\hat{\imath},\hat{\jmath}).
\end{aligned}$$

**Observation 2.** Given $j, \hat{\jmath} \in [1 \dots H]$, $\bar{W}^p(i,j)/\bar{W}^p(i,\hat{\jmath})$ is a constant for any $i \in [1 \dots N]$.

*Proof.*
$$\begin{aligned}
&\bar{W}^p(i,j)/\bar{W}^p(i,\hat{\jmath})\\
&= f^i \cdot \textstyle\sum_{j'=1}^{H} \left[w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'} / f^i \cdot \textstyle\sum_{j'=1}^{H} \left[w_{\hat{\jmath},j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'}\\
&= \textstyle\sum_{j'=1}^{H} \left[w_{j,j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'} / \textstyle\sum_{j'=1}^{H} \left[w_{\hat{\jmath},j'}^{p\to p} + w_{j'}^{c\leftrightarrow p}\right] \lambda_{j'}\\
&= \bar{W}^p(\hat{\imath},j)/\bar{W}^p(\hat{\imath},\hat{\jmath}).
\end{aligned}$$

Note that the proxies are sorted in ascending order of cost $\bar{W}^p(1,j)$ and the videos are sorted in descending order of cost $\bar{W}^p(i,1)$ in the greedy algorithm, that is, $\bar{W}^p(i,j) \le \bar{W}^p(i,j')$ for $j \le j'$ and $\bar{W}^p(i,j) \ge \bar{W}^p(i',j)$ for $i \le i'$. We then have another two observations:

**Observation 3.** $\bar{W}^p(i,j) - \bar{W}^p(i,j-\hat{\jmath}) \le \bar{W}^p(i-\hat{\imath},j) - \bar{W}^p(i-\hat{\imath},j-\hat{\jmath})$ for $\hat{\imath} \in [1 \dots (i-1)]$ and $\hat{\jmath} \in [1 \dots (j-1)]$.

*Proof.* From observation 1, we have $W^p(i,j)/\bar{W}^p(i-\hat{\imath},j) = \bar{W}^p(i,j-\hat{\jmath})/\bar{W}^p(i-\hat{\imath},j-\hat{\jmath}) = a$, where $a$ is a constant. This is equivalent to $\bar{W}^p(i,j) = a \cdot \bar{W}^p(i-\hat{\imath},j)$ and $\bar{W}^p(i,j-\hat{\jmath}) = a \cdot \bar{W}^p(i-\hat{\imath},j-\hat{\jmath})$. Here, $0 \le a \le 1$ because $\bar{W}^p(i,j) \le \bar{W}^p(i-\hat{\imath},j)$ for $\hat{\imath} \ge 0$. It follows that

$$\begin{aligned}
&\bar{W}^p(i,j) - \bar{W}^p(i,j-\hat{\jmath})\\
&= a \cdot \bar{W}^p(i-\hat{\imath},j) - a \cdot \bar{W}^p(i-\hat{\imath},j-\hat{\jmath})\\
&\le \bar{W}^p(i-\hat{\imath},j) - \bar{W}^p(i-\hat{\imath},j-\hat{\jmath}).
\end{aligned}$$

**Observation 4.** $\bar{W}^p(i,j+\hat{\jmath}) - \bar{W}^p(i,j) \ge \bar{W}^p(i+\hat{\imath},j+\hat{\jmath}) - \bar{W}^p(i+\hat{\imath},j)$ for $\hat{\imath} \in [1 \dots (N-i)]$ and $\hat{\jmath} \in [1 \dots (H-j)]$.

*Proof.* From observation 1, we have $\bar{W}^p(i+\hat{\imath},j)/\bar{W}^p(i,j) = \bar{W}^p(i+\hat{\imath},j+\hat{\jmath})/\bar{W}^p(i,j+\hat{\jmath}) = b$, where $b$ is a constant. This is equivalent to $\bar{W}^p(i+\hat{\imath},j) =$

$b \cdot \bar{W}^p(i,j)$ and $\bar{W}^p(i+\hat{\imath}, j+\hat{\jmath}) = b \cdot \bar{W}^p(i, j+\hat{\jmath})$. Here, $0 \le b \le 1$ because $\bar{W}^p(i+\hat{\imath}, j) \le \bar{W}^p(i,j)$ for $\hat{\imath} \ge 0$. It follows that

$$\begin{aligned} &\bar{W}^p(i,j+\hat{\jmath}) - \bar{W}^p(i,j) \\ \ge\, & b \cdot \bar{W}^p(i,j+\hat{\jmath}) - b \cdot \bar{W}^p(i,j) \\ =\, & \bar{W}^p(i+\hat{\imath}, j+\hat{\jmath}) - \bar{W}^p(i+\hat{\imath}, j) \end{aligned}$$

The above two observations imply that swapping one unit data of video $i$ in proxy $j$ with that of video $i'$ ($i' \in [1 \ldots (i-1)]$) in proxy $j'$ ($j' \in [1 \ldots (j-1)]$) yields the same or higher total cost, and, similarly, swapping one unit data of video $i$ in proxy $j$ with that of video $i'$ ($i' \in [(i+1) \ldots N]$) in proxy $j'$ ($j' \in [(j+1) \ldots H]$) yields the same or higher cost. As the prefixes are fully packed to the proxies and there is no space left, the solution given by the greedy algorithm is optimal.

$\square$

# Bibliography

[1] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Proceedings of the 9th International Conference on Cooperative Information Systems table of contents (CoopIS 2001)*, 2001.

[2] S. Acharya and B. Smith. Middleman: A Video Caching Proxy Server. In *Proceeding of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'00)*, June 2000.

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

[4] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. On Optimal Batching Policies for Video-on-Demand Storage Servers. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'96)*, June 1996.

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

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

[7] BitTorrent. http://www.bittorrent.com.

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

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

[10] Y. Chae, K. Guo, M. M. Buddhikot, S. Suri, and E. W. Zegura. Silo, Rainbow, and Caching Token: Schemes for Scalable, Fault Tolerant Stream Caching. *IEEE Journal on Selected Areas in Communications*, 20(7):1328–1344, September 2002.

[11] Y. Chawathe, S. McCanne, and E. Brewer. An Architecture for Internet Content Distribution as an Infrastructure Service. *http://www.cs.berkeley.edu/yatin/papers/scattercast.ps.*

[12] S. Chen, B. Shen, S. Wee, and X. Zhang. Designs of High Quality Streaming Proxy Systems. In *Proceedings of the IEEE INFOCOM 2004*, Hong Kong, March 2004.

[13] S. Chen, B. Shen, Y. Yan, S. Basu, and X. Zhang. SRB: Shared Running Buffers in Proxy to Exploit Memory Locality of Multiple Streaming Media Sessions. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan, March 2004.

[14] Y. Cui, B. Li, and K. Nahrstedt. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[15] G. B. Dantzig. Linear Programming and Extensions. *Princeton University Press*, 1963.

[16] S. G. Dykes and K. A. Robbins. A Viability Analysis of Cooperative Proxy Caching. In *Proceedings of the IEEE INFOCOM 2001*, April 2001.

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

[18] H. Garcia-Molina. Elections in A Distributed Computing System. *IEEE Transactions on Computers*, 31(1):48–59, January 1982.

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

[20] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. PROP: A Scalable and Reliable P2P Assisted Proxy Streaming System. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan, March 2004.

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

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

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

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

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

[26] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: Peer-to-peer Media Streaming using CollectCast. In *Proceedings of the ACM Multimedia*, November 2003.

[27] M. M. Hefeeda, B. K. Bhargava, and D. K.-Y. Yau. A Hybrid Architecture for Cost-Effective On-Demand Media Streaming. *Computer Networks*, 44(3):353–382, February 2004.

[28] M. Hofmann, T. E. Ng, K. Guo, S. Paul, and H. Zhang. Caching Techniques for Streaming Multimedia over the Internet. *Technical Report*, April 1999, Bell Labs.

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

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

[31] N. Katoh, T. Ibaraki, and H. Mine. Notes on the Problem of the Allocation of Resources to Activities in Discrete Quantities. *Journal of Operational Research Society*, 31:595–598, 1980.

[32] Kazaa. http://www.kazaa.com.

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

[34] E. Kusmierek, Y. Dong, and D. H.-C. Du. Loopback: exploiting collaborative caches for large-scale streaming. In *Proceedings of the Twelfth Annual Multimedia Computing and Networking (MMCN'05)*, California, USA, 2005.

[35] J. Liu and J. Xu. Proxy Caching for Media Streaming over the Internet. *IEEE Communications*, August 2004. (to appear).

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

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

[38] Z. Miao and A. Ortega. Scalable Proxy Caching of Video Under Storage Constraints. *IEEE Journal on Selected Areas in Communications*, 20(7):1315–1327, September 2002.

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

[40] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. In *Proceeding of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02)*, May 2002.

[41] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster. 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*, 2003.

[42] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of the IEEE INFOCOM 1999*, New York, NY, March 1999.

[43] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *Proceedings of the ACM SIGCOMM'02*, August 2002.

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

[45] D. A. Tran, K. A. Hua, and T. Do. ZIGZAG: An Efficient Peer-to-peer Scheme for Media Streaming. In *Proceedings of the IEEE INFOCOM 2003*, San Francisco, CA, USA, April 2003.

[46] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In *Proceedings of the IEEE INFOCOM 2002*, New York, June 2002.

[47] J. Wang. A Survey of Web Caching Schemes for the Internet. *ACM Computer Communication Review (CCR)*, 29(5), October 1999.

[48] W. Wang and B. Li. 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).*, 2003.

[49] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the Scale and Performance of Cooperative Web Proxy Caching. In *Proceedings of the SOSP'99*, December 1999.

[50] D. Wu, Y. T. Hou, and Y.-Q. Zhang. Transporting Real-time Video over the Internet: Challenges and Approaches. *Proceedings of the IEEE*, 88(12), December 2000.

[51] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-Based Proxy Caching of Multimedia Streams. In *Proceedings of the 10th international conference on World Wide Web (WWW-10)*, Hong Kong, May 2001.

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

[53] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of the IEEE INFOCOM 1996*, SF, CA, March 1996.

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

[55] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *Proceeding of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, NY, June 2001.

CUHK Libraries
004278899