SmartSharing: A Content Delivery Network with Local Sharing of Over-the-Top Devices

by

Jiamin Fan
B.Sc., Nanjing University of Posts and Telecommunications, 2014

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Supervisory Committee

Dr. Kui Wu, Supervisor
(Department of Computer Science)

Dr. Sudhakar Ganti, Departmental Member
(Department of Computer Science)

ABSTRACT

Content delivery networks (CDNs) depend on distributed cache servers to reduce the content delivery distance and latency to end users. Nevertheless, a CDN's cache footprint is greatly limited by the high cost in deploying and maintaining large-scale cache servers. To break the limit, CDN providers adopt a new content caching strategy that allows end users to share their storage/bandwidth resources with each other. Two core questions need to answer in this CDN strategy: (1) how to incentivize end users to contribute their resources? and (2) how to facilitate transparent, secure content exchange among end users?

We propose a new CDN solution, called SmartSharing, where users contribute their Over-the-top (OTT) devices as mini cache servers. With SmartSharing, an OTT device can share the content the OTT owner is downloading and in addition can cache content for neighboring OTT devices in the same area. To incentivize end users to contribute their resources, SmartSharing uses game theory and the Expectation-Maximization (EM) algorithm to determine content delivery schedule and the pricing scheme. To facilitate content trading among end users, SmartSharing uses smart contracts in Ethereum to create a transparent and safe transaction platform. We evaluate SmartSharing with real-world trace driven simulation as well as smart contract prototype in Ethereum using content meta-data and the derived pricing scheme. By disclosing the internal dynamics in content delivery schedule and pricing scheme and analyzing the overhead in content trading, we show that SmartSharing is an effective new CDN solution that benefits content providers, CDN, and end users.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Supervisor, Dr. Kui Wu, for mentoring, support, guidance, encouragement, and patience.

Drs. Sudhakar Ganti and Hong-chuan Yang for their willingness to be on this committee.

Dr. Daming Liu for his time and the discussions that we had.

UVic friends and office mates, Huan Wang, Guoming Tang for good advices.

My family for supporting me in the low moments.

Thank you.
Jiamin Fan

DEDICATION

To my family

Chapter 1

Introduction

1.1 What Is Content Delivery Network?

With the rapid development of the Internet, the number of network applications involving high volume of multimedia traffic has increased quickly in the past years. Multimedia traffic includes different types of media content, such as images, audio, and video. Video content alone has occupied a large portion of Internet bandwidth, increased from 64% in 2014 to 84% in 2019 according to the survey of Cisco. The massive amount of multimedia Internet traffic may cause network congestion and longer delay in content delivery, resulting in profit loss for Internet service providers. According to a report[1], a page load slowdown of just one second could cost it $1.6 billion in sales each year." To improve the service quality (bandwidth, delay, packet loss) of Internet applications, researchers from both academic and industry are looking for effective solutions.

Content delivery networks (CDNs) have been used to speed up contend delivery and have become an indispensable component in the whole Internet ecosystem. CDN users can obtain service from the nearby cache servers since the content is replicated to these cache servers in advance. As shown in Fig. 1.1, a CDN extensively uses a lot of edge cache servers to cache content [9] at the network edge, and by doing so, they significantly reduce the traffic delivery distance and the latency from service providers to users. Cache servers are dedicated servers used to store web content. They are deployed in places geographically close to end users and mitigate the toughest challenges of delivering content over the Internet.

[1] `https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales`

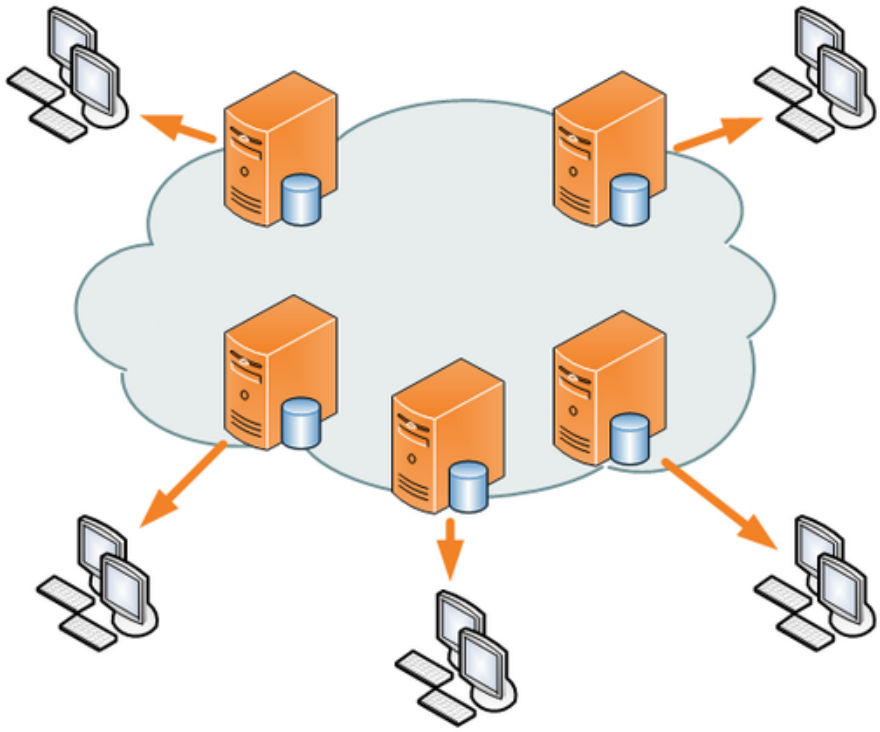


Figure 1.1: CDN uses cache servers close to end users for content delivery

CDNs have many advantages. They can reduce the bandwidth costs, improve access times to data and increase global availability of content. With CDNs, customers can enjoy a better quality of experience. Happy customers also imply high business profit to service providers.

Due to these advantages, CDN market has seen a sharp rise in the past few years and is expected to further grow up to 23.2 billion US dollars by 2021 [15]. High profits of CDN market have attracted more and more companies to join the competition in the past few years, evidenced by several examples as follows:

- Akamai: One of the largest and most popular content delivery network companies in the world. The company comprises of two hundred thousand servers around the globe. It also offers cloud-based high definition (HD) video streaming.

- Amazon CloudFront: The content delivery network of Amazon web services. It works together with Amazon Cloud Storage (S3) and Amazon Elastic Cloud

(EC2) services. CloudFront provides an easy way for users to store their content.

- CDN77: Founded in 2012, CDN77 provides fair, transparent and flexible CDN solution to customers. It is headquartered in London and has 28 data centers across 23 countries.

- CloudFlare: Founded in 2009, it focuses on the business of web acceleration and security. Customers can optimize their websites by using CloudFlare CDN service and can protect their websites by using CloudFlare security suite.

Due to the high competition, CDN providers need to further expand their footprint, i.e., the number of cache servers in their points of presence (PoPs) to maintain their advantages. For instance, Akamai, one of the most popular CDN providers, owns more than two hundred thousand cache servers over fifteen thousand ISPs around the world. In order to compete for market share, CDN companies are madly spreading their footprint and deploying more and more cache servers. The expansion of CDN footprint, however, is limited by the capacity of Internet backbone. In addition, the investment of a large number of duplicate nodes would eat up a large portion of the profit from the CDN provider. Deploying a higher number of cache servers incurs higher equipment cost and heavier maintenance overhead for the CDN providers. Such a large cost on equipment and maintenance makes small companies unable to compete with large capital-rich companies, resulting in unfair competition in the CDN market.

Another problem of deploying more edge cache servers is that the CDN providers will has less funds for technology research and development, which is not conducive to the sustainable development for the CDN market. Even worse, the high cost of edge servers eventually will be passed on to the users, leading to a high price for CDN services. As shown in Table 1.1, the current cost for CDN services is nontrivial.

Table 1.1: Example CDN prices

Monthly costs	AKAMAI	CDN77	CLOUDFLARE	MAXCDN	STACKPATH
6 TB plan	$900	$230	$500	$369	$260
25 TB plan	$8000	$500	$1500	$1164	$600
100 TB plan	$8000	$1890	$3000	$4539	$2100
Minimum contract term	12 Months	1 Month	12 Month	1 Month	1 Month

To alleviate the problem, some researchers proposed the concept of CDN federation, i.e., CDN companies cooperate with each other and share their cache servers to the federation. However, the implementation of CDN federation may cause new problems, such as the difficulties in profit distribution and resource management. As such, in this thesis, we ignore CDN federation, even if it has been actively pushed forward in the CDN industry.

1.2 Motivation and Contributions

In order to overcome the technical and economic difficulties of deploying edge servers, we take the opportunity from the surge of over-the-top (OTT) devices at the premises of end users. OTT devices are deployed by content providers (CPs) that distribute streaming media as a standalone product directly to users over the Internet. Our key idea is to utilize idle OTT devices as content caches and ask OTT devices in the same area to help each other. The key to its success is to *incentivize* and *facilitate* ordinary Internet users to contribute their OTT devices for content cache and the idle bandwidth resource for content delivery.

To *incentivize* end users to contribute their OTT devices and idle resources, the majority of CDN providers in the current CDN market give virtual credits to end users as a reward, which can be monetized in the form of goods, prizes, or free services. Nevertheless, it is extremely challenging to design such a centralized (virtual) reward mechanism [4], because the actual value of virtual credits is hard to determine in advance. More often than not, such type of reward systems puts the CDN provider in a high risk of profit losses if virtual credits are over valued to motivate end users' participation.

To *facilitate* end users to share their OTT storage and bandwidth resources, we need to support automatic resource trading without worrying about flawed transactions. Since end users are on a free market lacking of enough mutual trust, they need to find a way for transparent, secure, smooth trading among them without any middlemen.

In this thesis, we adopt an unified framework, called SmartSharing, with the aim of addressing the above two problems. To solve the former problem, we propose a new incentive and pricing scheme to encourage OTT owners to contribute their resource and devices. This scheme completely offloads the economic risks from the CDN providers. The key idea of this framework is to ask OTT owners to purchase content

from nearby OTT owners and let them compensate each other based on their own cost and services contributed to the SmartSharing system. To gain more customers, OTT owners need to compete in a well-designed pricing game [13] [3] [14]. In this way, the CDN provider not only avoids the economic risks of rewarding virtual credits to contributors, but benefits from the expansion of CDN cache footprint. In other words, the profits of CDN providers in SmartSharing are completely guaranteed and protected by this pricing scheme. In addition, the benefit of OTT owners are also guaranteed since the pricing scheme is developed under the constraint that fetching cached content from other OTT owners must be cheaper than fetching the content directly from the Internet service provider. To solve the latter problem, we use smart contracts [11] in Ethereum [19] to provide an automatic and secure trading platform for end users. A trading website is designed for OTT owners to purchase and sell content. A wallet is allocated to each OTT owner to support the function of deploying smart contracts and paying with virtual currency.

The contributions of the thesis include:

- First, in Chapter 2, we design a new CDN framework, called SmartSharing, where OTT owners contribute their OTT devices as caching servers. Compared to existing peer-to-peer CDN and crowdsourcing CDN, SmartSharing is totally different. Hence, SmartSharing requires a new set of pricing solutions and content fetching solutions. For CDN providers, SmartSharing can extend their footprint and deliver content more quickly since OTT devices are contributed as mini cache servers by OTT owners. CDN providers can obtain such a huge benefit without the need to deploy new cache servers or take any risk of issuing virtual credits to OTT owners who contribute the resources.

- Second, we solve the two key technical challenges in SmartSharing: (a) incentivizing OTT owners to contribute and (b) facilitating content trading among OTT owners. For the former, in Chapter 3, we design a content delivery scheduling strategy for the OTT owners based on the Nash equilibrium in a bandwidth game that maximizes each OTT owner's payoff, and then develop a pricing scheme that aligns with the Nash equilibrium solution. If multiple OTT devices host the same content, the OTT device with a higher Nash equilibrium contribution value is more likely to be selected as the provider. In other words, contributors can gain more benefit if they contribute more resource to the system. The pricing scheme of SmartSharing is designed to fit the content delivery

scheme derived from the bandwidth game. For the latter, in Chapter 4, we develop a decentralized content trading platform with smart contract [11] in Ethereum [19]. OTT owners in SmartSharing can buy and sell content over this transparent and flexible third-party trading platform.

- Third, in Chapter 5, with trace-driven simulation using different system parameter settings, we thoroughly disclose the dynamics of bandwidth game and the pricing scheme in SmartSharing. We evaluate the speed and the cost of executing smart contracts for content trading. Our results show that the overhead in terms of delay and monetary cost for executing smart contracts in SmartSharing is very small.

1.3 Related Work

Related work can be roughly divided into two categories: crowsourcing CDN [20] [21] and peer-to-peer CDN [1, 10]. The first one includes techniques that recruit ordinary Internet users with idle resources to worked as caching servers. The second category includes techniques that create a distributed network of computing resources, consisting of users' idle equipment. To better understand related work, we not only introduce some academic research, but also introduce several successful industrial projects.

1.3.1 Crowdsourcing CDN

In the category of crowdsourcing CDN, any ordinary Internet users with idle resources can be recruited by crowdsourcing as caching servers. With the decline of storage cost, ordinary users' equipment is likely to have a high amount of free storage space. Similarly, users' bandwidth may also not be fully occupied. If enough incentives are offered, these Internet users can contribute their storage resource for content caching and bandwidth resource for content delivery. In this way, crowdsourcing CDN benefits the CDN provider by reducing the expenditure of deploying and maintaining edge servers, and benefits the crowd contributors by providing them with monetary rewards or service credits.

For example, Thunder Crystal, the crowdsourcing CDN introduced in [21], can achieve about half price of traditional CDN services because of the use of cheaper crowd contributors' resources. Nevertheless, it has been reported that it is difficult to

motivate ordinary Internet users to contribute their resources if CDN provider cannot offer them a sizable bonus, which would eat up the CDN profit.

1.3.2 Peer-to-Peer CDN

In the category of peer-to-peer CDN, advanced technology is used to create a distributed network of computing resources, consisting of users' idle equipment such as TV boxes, iPad, and cell phones. BlockCDN [1] is a special case of peer-to-peer CDN, which is in particular related to our work. BlockCDN is a new type of CDN model built on Ethereum smart contracts [11]. Built on smart contract which facilitates distributed resource trading, blockCDN presents new business opportunities for CDN industry. Since the available resources of end users' idle equipment is volatile, the key to blockCDN's success is to design spontaneous pricing mechanism for supporting dynamic incentives for users to continuously contribute their resources. Nevertheless, the internal pricing and content scheduling schemes in existing BlockCDN [1] are unknown. SmartSharing focuses on OTT devices, which have more stable Internet connection and bandwidth resources, while BlockCDN [1] aims at more general devices, including cell phones, whose available storage and bandwidth resources may be volatile.

Chapter 2

System Model and Problems

2.1 System Model

The architecture of SmartSharing is shown in Fig. 2.1. There is an OTT device in each house. OTT devices in nearby neighborhood are grouped together, and OTT devices in the same group can serve each other. OTT devices have buffers to temporarily cache content, and can quickly deliver content to other OTT devices in the same group.

As an example to illustrate the benefit and operations of SmartSharing, we assume that there are three OTT owners in Area 1. Assume that time is slotted and at the second time slot, the content in each OTT buffer is as follows: OTT1 (c_1, c_2), OTT2 (d_1, c_1), OTT3 (d_1, d_3), where c_i and d_i denotes the index of content. Assume that the content requests at the next time slot are c_3 at OTT1, c_2 at OTT2, and c_1 at OTT3. We can see that for the next time slot, OTT1 can serve OTT2 and OTT3, and OTT1 and OTT2 can collectively serve OTT3 (i.e., OTT3 can fetch partial of c_1 from OTT1 and partial of c_1 from OTT2).

OTT devices are located at end users' premises and are used to receive/store media content delivered over the Internet. In addition, OTT devices are usually capable of simple computation. As such, an OTT device could be treated as a mini edge server, which serves its owner as well as other nearby users.

In this paper, we assume that

- OTT devices are grouped by different local areas, and only OTT devices in the same area can serve each other. This assumption is made for two reasons: (1) For the consideration of both performance and cost, traffic among nearby OTTs

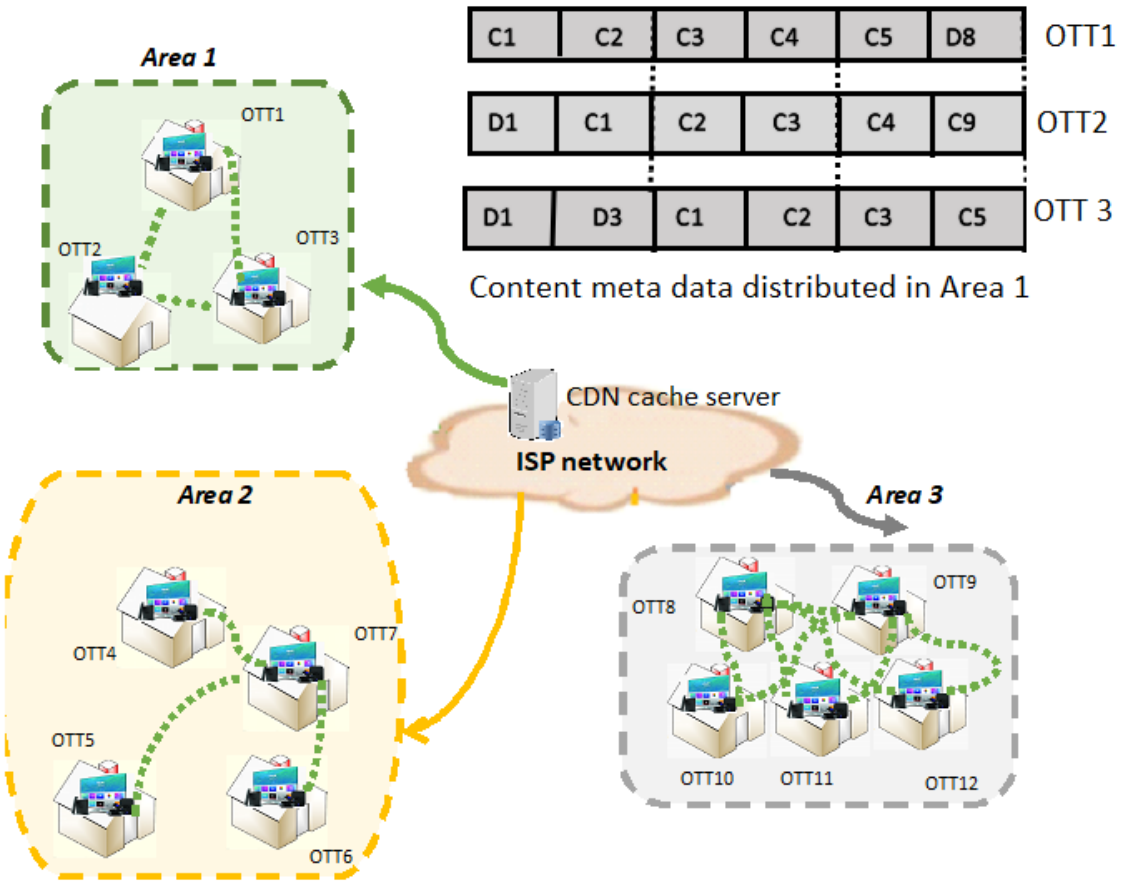


Figure 2.1: Architecture of SmartSharing

has smaller delay and uses much less network bandwidth than traffic to remote OTTs. Content exchange among nearby OTTs can be easily handled with local wifi or over local Ethernet segments; (2) Our system requires an OTT device to share content meta data with other OTT devices in the same group, and as such restricting content sharing in local areas greatly simplifies system design and pricing mechanism.

- We assume that the OTT groups do not overlap. In other words, an OTT belongs to only one group. Allowing OTT devices to provide service for multiple groups is an interesting topic left as our future research.

- The OTT devices have enough processing power to execute smart contract primitives for service trading. This assumption is reasonable because blockchain

based smart contracts have been deployed and tested in Internet of Thing (IoT) environment [6], where the involving entities are small sensors that have much weaker processing power than OTT devices.

Note that SmartSharing may cause concern of copyright infringements, because multimedia content can be downloaded, stored temporarily, and shared to other OTTs. Nevertheless, such a concern exists in most peer-to-peer multimedia streaming systems [5] as well as TV streaming devices that connect to various media sources over the Internet, download and distribute files over the device. It is debatable whether or not these systems violate copyright laws [2], depending on detailed terms in a specific region. Thus, we ignore the copyright issues and only focus on the technical ingredients of SmartSharing.

2.2 Problems

Based on the SmartSharing system model we have introduced above, OTT devices in the same group can server each other. OTT devices store content in their buffers and can deliver content to other OTT owners. To implement the above system in practice, we need to address two technical problems:

(1) How to design a pricing scheme to incentivize users to contribute their OTT devices and resource to SmartSharing system?

In the current pricing schemes, contributors are rewarded by virtual credits. However, it is difficult to design such a centralized reward mechanism for OTT owners in advance. Even worse, it is possible for CDN providers to lose their profit if virtual credits are over valued to incentive contributors to contribute their OTT devices to SmartSharing system. We solve this problem with a combined approach of game theory and EM algorithm in Chapter 3. In our pricing scheme, OTT devices can compensate each other based on their cost and services contributed to the system.

(2) How to facilitate content trading among OTT devices?

Given a content delivery schedule and a pricing scheme, OTT owners can trade content following the delivery schedule and the pricing scheme. In a fully-distributed environment lack of enough mutual trust, we need a transparent and distributed trading platform for OTT owners to safely trade contents. We solve this problem with blockchain-based smart contracts in Chapter 4.

Chapter 3

Pricing in SmartSharing

Each OTT owner in SmartSharing has its own OTT device. OTT owners can download content from ISP in advance and then deliver their buffered content to other OTT owners. We design a pricing scheme in this chapter to set prices for OTT owners. This scheme is divided into three parts. In the first part, we formulate a bandwidth game [3] and calculate the Nash equilibrium contribution value for each OTT owner. In the second part, content delivery schedule is obtained based on the Nash equilibrium result in a bandwidth game. In the third part, we derive the price strategies for all the OTT owners. Normally, the price should be set in advance and then OTT owners decide their content delivery schedules. However, it is difficult to set an optimal price in advance because of the dynamic changes in content that could be shared among OTTs. We use the EM algorithm to help us find the suitable pricing scheme that align the best with our content delivery schedule. The above calculation is completed independently in each OTT owner's own equipment. Finally, the final price of each OTT owner will be published to the SmartSharing website through smart contract, which will be introduced in next chapter.

3.1 Bandwidth Game

Before we introduce the bandwidth game, we need define several concepts in SmartSharing.

Definition 1. ***Demand-supply matrix:*** *It is binary $n \times n$ matrix $Y = [y_{ij}]$ capturing the content demand and content supply among OTT devices, where n is the number of OTT devices in consideration, $y_{ij} = 1$ if OTT i has the content for OTT*

j's next request, and $y_{ij} = 0$ otherwise.

Definition 2. ***Contribution:*** *We assume that an OTT device has the freedom to determine whether or not and how much resource (e.g., bandwidth and storage[1]) it would contribute. To fairly evaluate the contribution, we assume that all OTT devices have the same amount of resource and denote the contribution from an OTT as the percentage of its resource that the OTT would contribute.*

Definition 3. ***Benefit matrix:*** *When an OTT device benefits from the services of other OTT devices, we encode this benefit using an $n \times n$ matrix $I = [I_{ij}]$, where n is the number of OTT devices, I_{ij} denotes the value that the contribution from OTT i is worth to OTT j. Set $I_{ij} = 0$ if OTT j is not interested in OTT i's contribution.*

We assume a linear dependence between contribution and the benefit. To be specific, we assume $I_{ij} = k_j y_{ij} d_i$, where k_j is a weight associated with the utility of OTT j and d_i denotes the contribution from OTT i. Note that k_i means how much utility that an OTT owner can benefit from the unit contribution of the other OTT owners. Different OTT owners have different values of k_i. The practical meaning is that when OTT i has the content of OTT j, the higher the resource amount (implying higher network bandwidth or higher I/O speed) that OTT i contributes, the higher the value this contribution to OTT j.

The players of bandwidth game are the set of all OTT devices, denoted by N_r. The strategy $d_i (0 \leq d_i \leq 1)$ of player $i \in N_r$ is OTT i's contribution to SmartSharing. Since the payoff of player i is a function of both his own strategy d_i and other players' strategies denoted by d_{-i}, we denote the payoff of player i as $r_i(d_i, d_{-i})$

Definition 4. ***Bandwidth game:***

- *Players: The set of all OTT devices, denoted by N_r.*
- *Strategies: OTT i's contribution $d_i (0 \leq d_i \leq 1)$ to SmartSharing.*
- *Payoffs: $r_i(d_i, d_{-i})$*

[1]We use the general term resource without distinguishing different resource types, since the same model can be used for different resources.

3.2 Solutions to the Bandwidth Game

3.2.1 Preliminary Analysis

To solve the bandwidth game, we need a rational assumption: the higher the contribution from an OTT, the higher the probability that the OTT's content request should be satisfied by other OTT devices (if they have the content in cache). This assumption is to incentivize OTT devices to make contributions, because no OTT should be altruistic. As such, we adopt the following function to calculate the probability that OTT i's content request should be satisfied by other OTT devices, denoted by p_i:

$$p_i = \ln(1 + d_i), \tag{3.1}$$

where d_i is the contribution of OTT i to SmartSharing.

The payoff is equal to the utility minus the payment. Considering that OTT i can benefit from other OTT devices' contributions, the utility of OTT i can be calculated as:

$$u_i = \ln(1 + d_i) \sum_{j \neq i} I_{ji} = \ln(1 + d_i) \sum_{j \neq i} k_i y_{ji} d_j. \tag{3.2}$$

In the bandwidth game, the payment of OTT i is the fee charged by the ISP. The payment of OTT i is cd_i, where d_i denotes its bandwidth contribution and c is the unit bandwidth cost charged by the ISP.

The payoff function thus can be calculated as follows:

$$r_i = \ln(1 + d_i) \sum_{j \neq i} k_i y_{ji} d_j - cd_i \tag{3.3}$$

Definition 5. ***Nash equilibrium solution:*** *The Nash equilibrium is a solution concept of a non-cooperative game involving two or more players where each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy.*

In order to find the game's Nash equilibrium, each OTT is aimed at maximizing its own payoff, by selecting its best strategy when other OTT's strategies are given. In other words, we can obtain the Nash equilibrium for OTT i by setting the derivative of r_i w.r.t. d_i to 0.

$$r_i^{'} = \frac{1}{(1+d_i)} \sum_{j \neq i} k_i y_{ji} d_j - c = 0. \tag{3.4}$$

Solving (3.4), we obtain:

$$d_i = \frac{\sum_{j \neq i} k_i y_{ji} d_j}{c} - 1 \tag{3.5}$$

Since $0 \leq d_i \leq 1$,

$$d_i = min\left\{1, max\left\{\frac{\sum_{j \neq i} k_i y_{ji} d_j}{c} - 1, 0\right\}\right\}. \tag{3.6}$$

To ease notation, we denote the function of (3.6) as

$$d_i = T(d_{-i}). \tag{3.7}$$

When the solutions to all the d_i are established simultaneously, a Nash equilibrium solution of the bandwidth game is obtained. We next prove that the bandwidth game exists at least one Nash equilibrium.

Theorem 1. *There exists at least one Nash equilibrium in the bandwidth game.*

Proof. We first introduce the Brouwer's fixed-point theorem [16]: Every continuous mapping f from a compact convex set to itself has a fixed point z_0 satisfying $z_0 = f(z_0)$.

The best response mapping function $d_i = min\left\{1, max\left\{\frac{\sum_{j \neq i} k_i y_{ji} d_j}{c} - 1, 0\right\}\right\}$ is a continuous mapping from a compact convex set $[0, 1]$ to itself, which satisfies the requirement of the Brouwer fixed-point theorem. Hence, there exists a fixed point $\boldsymbol{d^*} = (d_1^*, d_2^*, \ldots, d_n^*)$ satisfying (3.5). The fix point $\boldsymbol{d^*}$ is the Nash equilibrium. In other words, there exists a Nash equilibrium solution for the bandwidth game. □

For two-player bandwidth game, a Nash equilibrium solution can be derived analytically. For multi-player bandwidth game, mathematically deriving all d_i values that simultaneously satisfy (3.6) is nontrivial. As such, we propose an iterative algorithm to obtain the solution.

3.2.2 Solution of Two-player Bandwidth Game

The players in the two-player bandwidth game are OTT 1 and OTT 2. The strategies in this game are the contributions d_1, d_2. The payoff of each OTT owner i is given by (3.3), i.e.,

$$r_1 = \ln(1 + d_1)k_1 y_{21} d_2 - c d_1 \tag{3.8}$$

$$r_2 = \ln(1 + d_2)k_2 y_{12} d_1 - c d_2. \tag{3.9}$$

Based on previous analysis, the best strategies for OTT 1 and OTT 2, denoted as d_1^* and d_2^*, respectively, satisfy:

$$d_1^* = \frac{k_1 y_{21} d_2^*}{c} - 1 \tag{3.10}$$

$$d_2^* = \frac{k_2 y_{12} d_1^*}{c} - 1. \tag{3.11}$$

Solving the above system of equations, we obtain:

$$d_1^* = \frac{c^2 + k_1 y_{21} c}{(k_1 y_{21} k_2 y_{12} - c^2)} \tag{3.12}$$

$$d_2^* = \frac{c^2 + k_2 y_{12} c}{(k_1 y_{21} k_2 y_{12} - c^2)}. \tag{3.13}$$

In addition, (3.12) and (3.13) need to meet the constraints in (3.6). As such, the final solution is adjusted as follows.

Case A: If $\frac{k_1 y_{21} d_2^*}{c} - 1 \leq 0$, then $d_1^* = d_2^* = 0$, meaning that none of the OTT devices is willing to contribute because their payoff is not positive.

Case B: If $\frac{k_1 y_{21} d_2^*}{c} - 1 \geq 1$, then $d_1^* = 1$ and

$$d_2^* = \begin{cases} \frac{k_2 y_{12}}{c} - 1 & \text{if } 1 < \frac{k_2 y_{12}}{c} < 2 \\ 1 & \text{if } \frac{k_2 y_{12}}{c} \geq 2 \\ 0 & \text{if } \frac{k_2 y_{12}}{c} \leq 1. \end{cases}$$

Case C: If $0 < \frac{k_1 y_{21} d_2^*}{c} - 1 < 1$, then:

- If $0 < \frac{k_2 y_{12} d_1^*}{c} - 1 < 1$, then

$$d_1^* = \frac{c^2 + k_1 y_{21} c}{(k_1 y_{21} k_2 y_{12} - c^2)}$$
$$d_2^* = \frac{c^2 + k_2 y_{12} c}{(k_1 y_{21} k_2 y_{12} - c^2)}.$$

- If $\frac{k_2 y_{12} d_1}{c} - 1 \leq 0$, then $d_1^* = 0, d_2^* = 0$.

- If $\frac{k_2 y_{12} d_1^*}{c} - 1 \geq 1$, then $d_2^* = 1$ and

$$d_1^* = \begin{cases} \frac{k_1 y_{21}}{c} - 1 & \text{if } 1 < \frac{k_1 y_{21}}{c} < 2 \\ 1 & \text{if } \frac{k_1 y_{21}}{c} \geq 2 \\ 0 & \text{if } \frac{k_1 y_{21}}{c} \leq 1. \end{cases}$$

3.2.3 Solution of Multi-player Bandwidth Game

For multi-player bandwidth game, however, mathematically deriving all d_i values that simultaneously satisfy (3.6) is nontrivial, and as such we propose an iterative algorithm, called Best Response Update Algorithm (BRUA), to find the solution. The idea of BRUA is simple: we randomly select an initial contribution for each d_i and then use (3.7) to update d_i values, round by round, until all d_i values converge (i.e., each d_i does not changes much in consecutive iterations). The detail of BRUA is illustrated in Algorithm 1.

Algorithm 1 Best Response Update Algorithm (BRUA)

Require: Initial contribution values $d_i^0 (i \in N_r)$, constant k_i associated with OTT i, small threshold ϵ, the supply-demand relationship matrix $[\mathbf{Y}_{ij}]$

Ensure: all d_i^* values

```
t=0 and Flag=0.
while Flag==0 do
    Calculate d_i^{t+1} = T(d_{-i}^t) for all i ∈ N_r.
    if |d_i^{t+1} − d_i^t| ≤ ε for all i ∈ N_r then
        Flag=1.
    end if
    t++.
end while
Return d_i^* = d_i^n for all i ∈ N_r.
```

In the following, we give the condition under which the Nash equilibrium of the bandwidth game is unique and BRUA is guaranteed to converge to the unique Nash equilibrium.

We first introduce the concept of contraction mapping.

Definition 6. ***Contraction mapping:*** *Let (X, p) be the distance space, and T be the mapping from X to X. If there exists a number $a(0 < a < 1)$ that makes $p(Tx, Ty) \leq a * p(x, y)$ for all $x, y \in X$, then T is called a contraction mapping.*

Theorem 2. *If $\forall i \neq j, \frac{k_i y_{ji}}{c*(n-1)} < 1$ holds, then the bandwidth game has unique Nash equilibrium and BRUA converges to this unique Nash equilibrium.*

Proof. The Luenberger's theorem [12] shows that every continuous mapping f from a compact convex set to itself has a unique fixed point z_0 satisfying $z_0 = f(z_0)$ if the mapping is contractive. For all i, we have

$$\begin{aligned}
&|T(d_i^*) - T(d_i)| \\
=&|(\frac{\sum_{j\neq i} k_i y_{ji} d_j^*}{c} - 1) - (\frac{\sum_{j\neq i} k_i y_{ji} d_j}{c} - 1)| \\
=&|(\frac{\sum_{j\neq i} k_i y_{ji} d_j^*}{c}) - (\frac{\sum_{j\neq i} k_i y_{ji} d_j}{c})| \\
=&\frac{\sum_{j\neq i} k_i y_{ji} |d_j^* - d_j|}{c}.
\end{aligned}$$

Denote $||.||$ as the vector norm. Then,

$$\begin{aligned}
&||T(\boldsymbol{d^*}) - T(\boldsymbol{d})|| \\
=&\sum_{i=1}^{n} |T(d_i^*) - T(d_i)| \\
=&\sum_{i=1}^{n} \frac{\sum_{j\neq i} k_i y_{ji} |d_j^* - d_j|}{c}
\end{aligned}$$

Assuming that $\forall i \neq j, \frac{k_i y_{ji}}{c*(n-1)} < 1$ holds, we have $\forall i, \frac{\sum_{j\neq i} k_i y_{ji}}{c} < 1$. Denote $\alpha = \max\{\frac{\sum_{j\neq i} k_i y_{ji}}{c}\}$ over all i. We have

$$||T(\boldsymbol{d^*}) - T(\boldsymbol{d})|| \leq \alpha ||\boldsymbol{d^*} - \boldsymbol{d}||,$$

indicating that the mapping T in BRUA is contractive and thus the sequence generated by BRUA converges to the unique Nash equilibrium. □

3.3 Content Delivery Scheduling Among OTT devices

Based on the Nash equilibrium result of bandwidth game in previous section, we then consider the content delivery scheduling. For this, we first build a fetch probability matrix.

Definition 7. ***Fetch probability matrix:*** *It is an $n \times n$ matrix $\boldsymbol{F} = [f_{ij}]$, where the element f_{ij} denotes the probability that OTT j fetches content from OTT i.*

By observing the supply-demand matrix, OTT j has the option to fetch the content from several OTT devices. Intuitively, a higher the contribution from an OTT, the higher the chance that other OTT devices should obtain content from that OTT. In other words, the probability that OTT j fetches content from OTT i is proportional to the contribution OTT i, d_i^*. As such, we calculate the values in the fetch probability matrix as:

$$f_{ij} = \frac{y_{ij} d_i^*}{g(\boldsymbol{d}_{-i}^*) + y_{ij} d_i^*}, \tag{3.14}$$

where $g(\boldsymbol{d}_{-i}^*) = \sum_{p \in N_r, p \neq i} y_{pj} d_p^*$ is the sum of the bandwidth of all the OTT devices (except OTT i) that can serve OTT j.

A content delivery schedule can be recorded by the content fetch matrix $\boldsymbol{X} = [x_{ij}]$, where $x_{ij} = 1$ means OTT j fetches content from OTT i, and $x_{ij} = 0$ otherwise. After we build the fetch probability matrix $\boldsymbol{F}$, a content delivery schedule can be easily created and recorded in the content fetch matrix as follows: we generate a random value a uniformly distributed in $[0, 1]$. If $a \leq f_{ij}$, set $x_{ij} = 1$; otherwise set $x_{ij} = 0$.

3.4 Pricing Scheme Based on EM Algorithm

After the content fetch schedule is determined, we then derive the OTT devices' pricing scheme. Since in practice it makes more sense to allow only a set of possible charging prices, we assume that a batch of K price choices are provided to OTT owners, and we use a vector $z(l), 1 \leq l \leq K$ to record the prices. Since an OTT has the option to fetch content either from the ISP directly or from other OTT devices, the price charged by OTT must be cheaper than that charged by the ISP, i.e., $z(l) < c$, where c is the bandwidth fee charged by ISP.

Table 3.1: Main notations

Term	*Definition*
t	Round of iteration
K	Number of choices for price
n	Number of all OTT devices
c	Bandwidth cost charged by ISP
z	Set of all price choices
x_{ij}	$x_{ij} = 1$ if OTT j fetches content from OTT i; $x_{ij} = 0$ otherwise
$\boldsymbol{X}_i$	OTT i's content delivery table
$\theta_{z(l)}$	The proportion of OTT devices that would accept price z(l)
$\boldsymbol{\theta}$	Set of proportion of OTT devices that would accept each price
$Q_{i,z(l)}$	The probability that OTT i chooses price $z(l)$
Q_i	OTT i's pricing scheme
$L(\boldsymbol{X}\|\boldsymbol{\theta})$	The likelihood function that $\boldsymbol{X}$ has all the observed values given $\boldsymbol{\theta}$

Definition 8. ***Pricing scheme:*** *The pricing scheme of OTT i is defined by a vector $Q_i = (Q_{i,z(1)}, Q_{i,z(1)}, \ldots, Q_{i,z(K)})$, where $\sum_{l=1}^{K} Q_{i,z(l)} = 1$, and $Q_{i,z(l)}$ denotes the probability that OTT i adopts price $z(l)$.*

Remark 1. *Since content trading among end users is a zero-sum game on cost, the traditional optimization strategy that maximizes the total saving does not apply here. Instead, the Nash equilibrium solution to the bandwidth game and the corresponding content delivery schedule consider the benefit of all users, and hence our goal is to find a pricing strategy that makes the content delivery schedule most likely to occur, i.e., given content delivery schedule $\boldsymbol{X}$ (observable), what is the pricing scheme that makes $\boldsymbol{X}$ most likely to happen?*

We adopt the Expectation-Maximization (EM) algorithm [8] to answer the above question. To find out the pricing scheme for each OTT, we introduce latent parameters $\boldsymbol{\theta} = (\theta_{z(1)}, \theta_{z(2)}, ...\theta_{z(K)})$, where $\theta_{z(l)}(1 \leq l \leq K)$ denotes the proportion of OTT devices that would accept price $z(l)$, or equivalently the probability that a given OTT would accept price $z(l)$. Initially, we randomly set $\boldsymbol{\theta}_{(0)} = (\theta_{z(1),(0)}, \theta_{z(2),(0)}, ...\theta_{z(K),(0)})$, where the subscript denotes the round of iterations in our following EM algorithm.

Given all values x_{ij} (i.e., the observable input), the EM algorithm iterates through the following two basic steps: E step and M step.

3.4.1 E Step

Given all values in $\boldsymbol{\theta}$, for each OTT i the probability that $\boldsymbol{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ has all the observed values at price $z(l)$ is:

$$p(\boldsymbol{X}_i|z(l), \theta_{z(l)}) = \prod_{j=1}^{n} (\theta_{z(k)})^{x_{ij}} * (1 - \theta_{z(k)})^{1-x_{ij}}. \tag{3.15}$$

The posterior probability of $z(l)$ given $\boldsymbol{X}_i$ and $\theta_{z(l)}$ is:

$$\begin{aligned} & p(z(l)|\boldsymbol{X}_i; \theta_{z(l)}) \\ & = \frac{\prod_{j=1}^{n} (\theta_{z(l)})^{x_{ij}} * (1 - \theta_{z(l)})^{1-x_{ij}}}{\sum_{k=1}^{K} \prod_{j=1}^{n} (\theta_{z(k)})^{x_{ij}} * (1 - \theta_{z(k)})^{1-x_{ij}}}. \end{aligned} \tag{3.16}$$

The likelihood function that $\boldsymbol{X}$ has all the observed values given $\boldsymbol{\theta}$ can be calculated as follows:

$$\begin{aligned} L(\boldsymbol{X}|\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log p(\boldsymbol{X}_i|\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \log \sum_{l=1}^{K} p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}). \end{aligned} \tag{3.17}$$

For each i, we denote its pricing scheme as Q_i, which is a distribution over $z(1), \ldots, z(K)$, i.e., $\sum_{l=1}^{K} Q_{i,z(l)} = 1, Q_{i,z(l)} \geq 0$. Consider the following:

$$\begin{aligned} & \sum_{i=1}^{n} \log \sum_{l=1}^{K} Q_{i,z(l)} \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}} \\ & \geq \sum_{i=1}^{n} \sum_{l=1}^{K} Q_{i,z(l)} \log \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}}. \end{aligned} \tag{3.18}$$

The last inequality is due to Jensen inequality [18].

Note that the function $\sum_{l=1}^{K} Q_{i,z(l)} \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}}$ is essentially the expectation of $\frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}}$. To obtain the equality in (3.18), we know it is sufficient that the expectation be taken

over a constant-valued random variable, i.e., we require $\frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}}$ be constant over all l's. Then $\frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}} = \frac{\sum_{k=1}^{K} p(\boldsymbol{X}_i, z(k)|\boldsymbol{\theta})}{\sum_{k=1}^{K} Q_{i,z(k)}} = \sum_{k=1}^{K} p(\boldsymbol{X}_i, z(k)|\boldsymbol{\theta})$. The last equality is because $\sum_{k=1}^{K} Q_{i,z(k)} = 1$. Therefore, the function $Q_{i,z(l)}$ at iteration t is:

$$\begin{aligned} Q_{i,z(l),(t)} =& \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}_{(t)})}{\sum_{k=1}^{K} p(\boldsymbol{X}_i, z(k)|\boldsymbol{\theta}_{(t)})} \\ =& \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}_{(t)}}{p(\boldsymbol{X}_i|\boldsymbol{\theta}_{(t)})} \\ =& p(z(l)|\boldsymbol{X}_i; \boldsymbol{\theta}^{(t)}) \\ =& \frac{\prod_{j=1}^{n} (\theta_{z(l),(t)})^{x_{ij}} (1-\theta_{z(l),(t)})^{1-x_{ij}}}{\sum_{k=1}^{K} \prod_{j=1}^{n} (\theta_{z(k),(t)})^{x_{ij}} (1-\theta_{z(k),(t)})^{1-x_{ij}}}. \end{aligned} \tag{3.19}$$

3.4.2 M Step

In this step, we try to find the values of $\boldsymbol{\theta}$ that maximize the joint log likelihood function $L(\boldsymbol{\theta}|\boldsymbol{X})$ with the fixed auxiliary Q_i obtained in the E step.

$$\begin{aligned} \boldsymbol{\theta} :=& \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{l=1}^{K} Q_{i,z(l)} \log \frac{p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})}{Q_{i,z(l)}} \\ =& \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{l=1}^{K} Q_{i,z(l)} (\log p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}) - \log Q_{i,z(l)}) \\ =& \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{l=1}^{K} Q_{i,z(l)} \log p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}) - Q_{i,z(l)} \log Q_{i,z(l)} \\ =& \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{l=1}^{K} Q_{i,z(l)} \log p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta}) \qquad (3.20) \\ =& \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{l=1}^{K} \log p(\boldsymbol{X}_i, z(l)|\boldsymbol{\theta})^{Q_{i,z(l)}} \\ =& \operatorname*{argmax}_{\theta} \prod_{i=1}^{n} \prod_{l=1}^{K} \theta_{z(l)}^{(Q_{i,z(l)} \sum_{j=1}^{n} x_{ij})} (1-\theta_{z(l)})^{(Q_{i,z(l)} \sum_{j=1}^{n} (1-x_{ij}))}. \qquad (3.21) \end{aligned}$$

Equality (3.20) is because $Q_{i,z(l)} \log Q_{i,z(l)}$ is constant. Equality (3.21) is because *log* is a monotonically increasing function. We can solve the above optimization problem by

taking derivatives over $\theta_{z(l)}$. The solution is denoted as $\boldsymbol{\theta}_{(t+1)} = (\theta_{z(1),(t+1)}, \theta_{z(2),(t+1)}, ..., \theta_{z(K),(t+1)})$, where

$$\begin{aligned}
&\theta_{z(l),(t+1)} \\
=&\frac{\sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n} x_{ij})}{\sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n} x_{ij}) + \sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n}(1 - x_{ij}))} \\
=&\frac{\sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n} x_{ij})}{\sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n} 1)} = \frac{\sum_{i=1}^{n}(Q_{i,z(l),(t)} \sum_{j=1}^{n} x_{ij})}{n \sum_{i=1}^{n} Q_{i,z(l),(t)}}.
\end{aligned} \tag{3.22}$$

The new $\boldsymbol{\theta}_{t+1}$ values will be used in the next-round E step. We iterate the E step and the M step as above until convergence, i.e., either the times of iteration reach a pre-defined threshold or the difference in learned parameters $\boldsymbol{\theta}$ between consecutive iterations fall within a given small threshold. The pricing strategy for each OTT is obtained from the convergence result of the last-round E step.

So far, we've obtained the different price options and all the OTT owners' price strategies through iterating the E step and the M step in our EM algorithm until convergence. Then we will introduce a smart contract-based content trading platform to execute the real transactions among OTT owners in the next chapter.

Chapter 4

Smart Contract-based Content Trading

4.1 A High-Level Introduction

With the content delivery schedule and pricing scheme determined, we implemented a prototype to facilitate content trading in SmartSharing. The prototype is based on smart contract over Ethereum [7]. Smart contract are self-executing contracts that are governed by the explicit terms and conditions laid out within them. The self-executing nature of these contracts provides a tremendous opportunity for applications that rely on data to drive transactions. The key information in SmartSharing contracts, mainly content, price, seller ID, and buyer ID, are obtained in previous section. Fig. 4.1 shows how prices are set in smart contracts.

With our prototype, an OTT owner can trade content with other OTTs and carry out the transactions in a decentralized blockchain platform without the need of a trusted authority or central server. The currency that an OTT owner uses for content trading is in the form of "ether", which is hold in each OTT owner's *Ethereum wallet*. Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform. Since ether is a too big unit, we define a smaller unit for virtual currency in SmartSharing, called transaction token (TT), and 1 ether = 200 TT. The Ethereum wallet interacts with decentralized applications on the Ethereum blockchain. It allows OTT owners to hold and secure ether built on Ethereum, as well as write, deploy and use smart contracts. Upon a successful transaction, the ether equivalent to a special price is transferred from the content buyer's Ethereum wallet to the content seller's

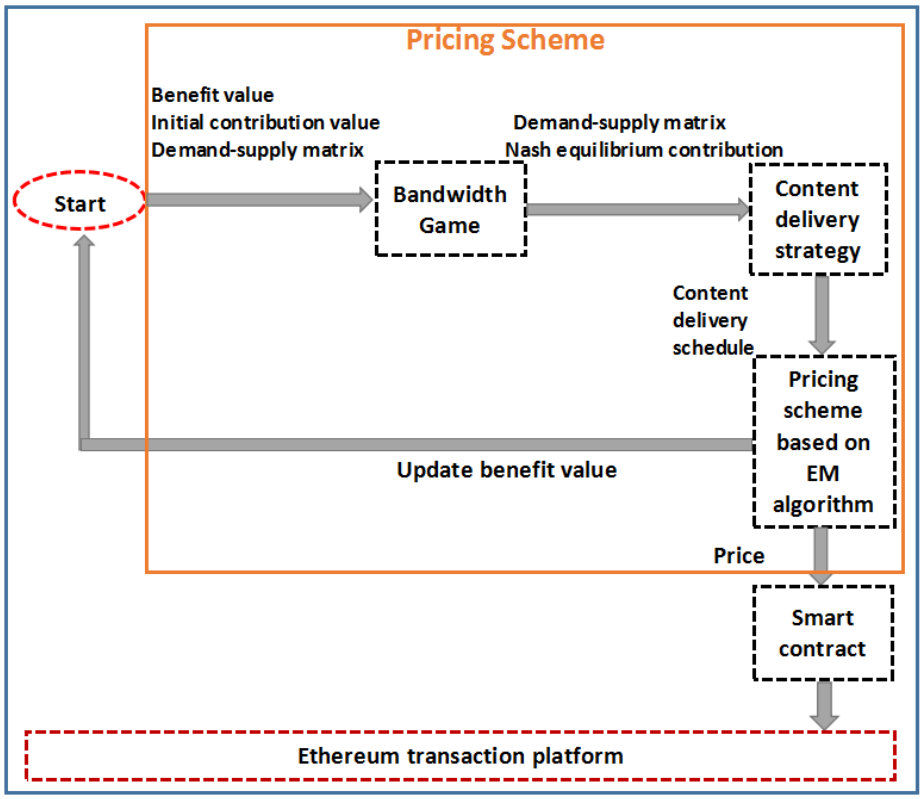


Figure 4.1: Setting prices in smart contracts

Ethereum wallet. We also build a web interface, as shown in Fig. 4.2, to help OTT owners trade content and manage their Ethereum wallet.

Smart contract-based content trading brings many benefits due to the inherent features of smart contracts over Ethereum [7]. The main advantages are listed as follows:

- **Transparency**

  The most important advantage of smart contracts is transparency. Smart contract's terms and conditions are completely visible to all the users. It is impossible for any person to alter them once the contract is established.

- **Security**

  Compared to a traditional contract, another important advantage of smart contract is security. Smart contracts can use the highest encryption level available in the world, which is the same standard that crypto-currencies use. This means

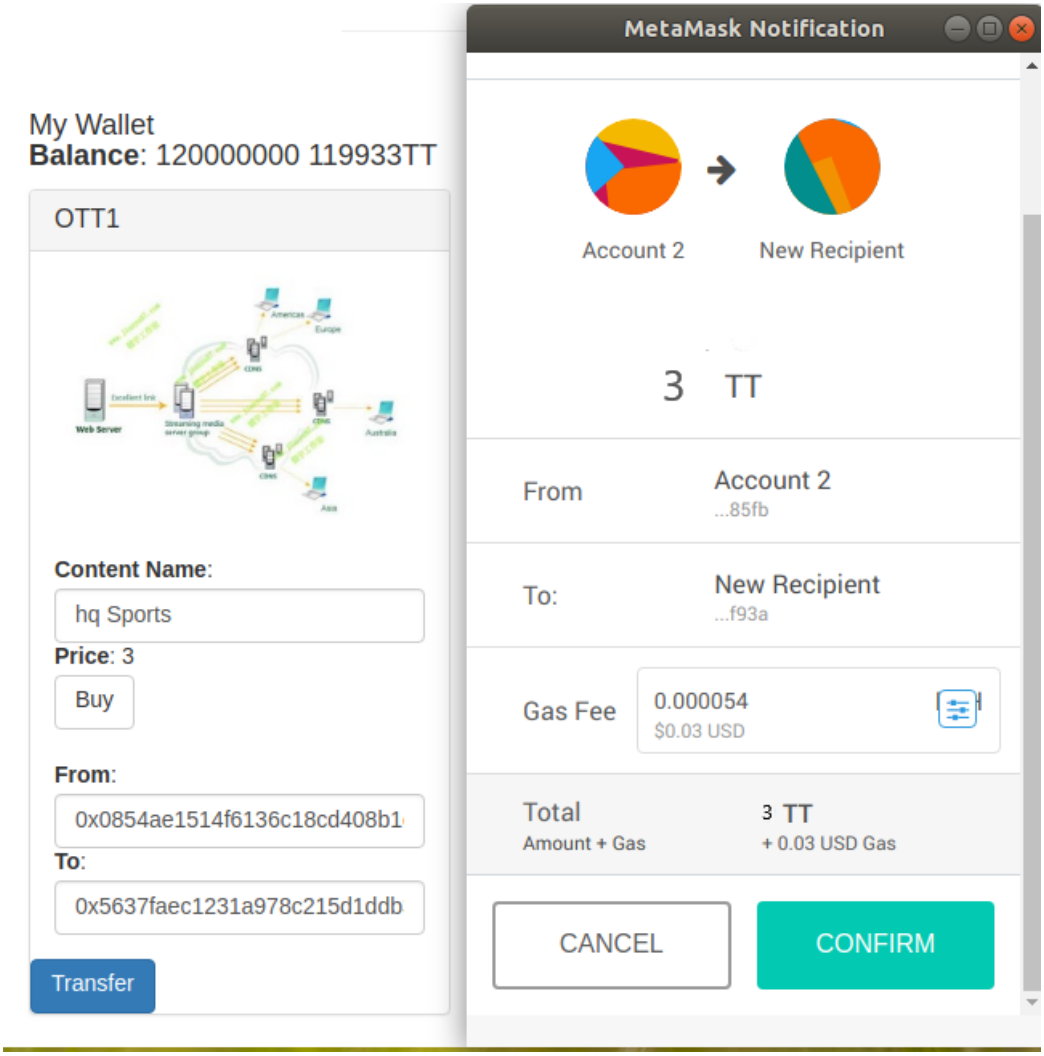


Figure 4.2: Interface for content trading and managing Ethereum wallet. TT is virtual currency defined in SmartSharing (1 ether = 200 TT, Gas is a measure defined in Ethereum for the amount of computational effort that it will take to execute operations in the transaction; gas fee depends on current market and are the fee that miners are willing to accept. The number 0.000054 is the gas fee in terms of ether, and 0.03 is the fee in terms of USD according to the market price on the day of June 20, 2018

that the high level protection of encryption makes smart contract the most secure web item. The attack to smart contract is practically impossible.

- **Efficiency**

  The verification node on the blockchain will verify the signature of the event

in advance in order to ensure the efficiency. If most verification nodes reach a consensus on the event, the smart contract will be successfully executed.

- **Accuracy**

  Explicit details of all terms and conditions are required to record in smart contract since an omission may cause transaction errors. As a result, automated contracts avoid the pitfalls of manually filling out heaps of forms.

The transparent, efficiency, and secure nature of the smart contracts significantly reduce the chances of manipulation or error during the transaction [19].

4.2 More Implementation Details

We provide the implementation details of SmartSharing trading platform in this section. The platform is driven by three smart contracts, recorded in *simpleStorage.sol*, *purchase.sol*, and $coinTransfer.sol$, respectively. *simpleStorage.sol* is used to push data into Ethereum; *purchase.sol* is used for purchasing content; $coinTransfer.sol$ is used to transfer virtual currency from the content buyer's Ethereum wallet to the content seller's Ethereum wallet.

4.2.1 The Key Information in Smart Contracts

Pricing scheme is the core algorithm of SmartSharing, which generates needed information for smart contracts. To save gas, smart contracts are executed by each OTT owner separately. The execution results will be pushed into Ethereum in the form of vectors through smart contract. To facilitate OTT users to trade easily, we also design a trading website and create an Ethereum wallet for each OTT owner.

Initial information of an OTT owner consists of content vector, benefit value, and initial contribution value. The content-supply vector that represents an OTT's content stored in its buffer is:

$$S_i := [s_i^1, s_i^2, ..., s_i^Q]^T$$

where $s_i^q = 1$ if OTT i has content q.

The initial information of each OTT owner is put into Ethereum through the *simpleStorage.sol* smart contract. A demand-supply matrix can be created by each OTT owner based on the initial information.

The initial contribution vector that represents all the OTTs initial contribution value is recorded in:

$$D^0 := [d_1^0, d_2^0, ..., d_n^0]^T$$

The demand-supply vector of each OTT i is represented as:

$$Y_j := [y_{1j}, y_{2j}, ..., y_{nj}]^T$$

where $y_{ij} = 1$ if OTT i has the content for OTT j's next request. Then Nash equilibrium contribution will be calculated by each OTT owner in the bandwidth game.

The equilibrium contribution vector that represents OTTs' Nash equilibrium contribution is written as:

$$D^* := [d_1^*, d_2^*, ..., d_n^*]^T$$

Demand-supply vector of each OTT owner and Nash equilibrium contribution vector are used as inputs to create content delivery schedule. After that, all the OTTs put their content delivery schedule into Ethereum through smart contract. Based on content delivery schedules of all the OTTs, a content fetch vector for each OTT is generated. This vector is used as an input of our EM algorithm. The output of EM algorithm are price schemes of each OTT. The purchase and content trading are then performed through *purchase.sol* contract and $coinTransfer.sol$ contract, respectively,

4.2.2 Data Structures

In this part, we list and explain the main data structures used in implementing the three smart contracts. These data structures are used by OTT owners to interactive with the blockchain platform. The second table introduces several useful contracts in smart-sharing.

4.2.3 User Interface

SmartSharing Website We build a website, which enables OTT owners to purchase and sell content in Smartsharing and carry out a transaction in a decentralized

Table 4.1: Main notations

Data Structure	*Definition*
s_i^q	$s_i^q = 1$ if OTT i has content q; $s_i^q = 0$ otherwise
S_i	The content-supply vector that represents OTT i's content
n	Number of all OTTs in smart-sharing system
d_i^0	Initial contribution value of OTT i
D^0	A contribution vector that represents OTTs initial contribution
y_{ij}	$y_{ij} = 1$ if OTT i has the content for OTT j's next request; $x_{ij} = 0$ otherwise
Y_j	The demand-supply vector of each OTT j

blockchain platform without the need of a trusted authority or central server. For OTTs who have contributed their devices to SmartSharing, they can view the real-time content supplied by other OTTs. As illustrated in Fig. 4.3, an user can check the content supplied by different OTTs through the website.

Ethereum Wallet Each OTT owner owns an Ethereum wallet. The Ethereum wallet is a gateway to decentralized applications on the Ethereum blockchain. It allows OTT owners to hold and secure ether and other crypto-assets built on Ethereum, as well as write, deploy and execute smart contracts. Once the block is successfully verified, the virtual currency equivalent to the agreed price is immediately displayed in the seller's Ethereum wallet and disappeared in the buyer's Ethereum wallet.

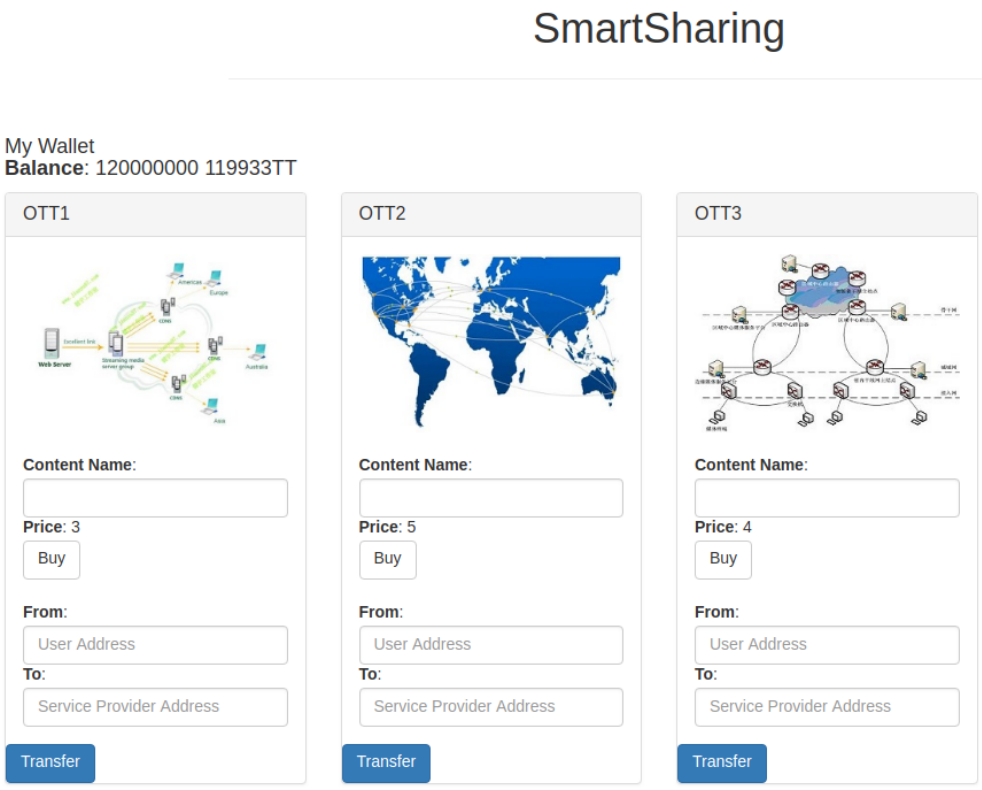


Figure 4.3: Transaction website

Chapter 5

Evaluation of SmartSharing

We first evaluate SmartSharing with trace-driven simulation to illustrate the details of bandwidth game and pricing scheme. We then evaluate the prototype with respect to the speed and cost of executing transactions. Together, trace-driven simulation and smart contract-based implementation tell a comprehensive picture on the performance of SmartSharing.

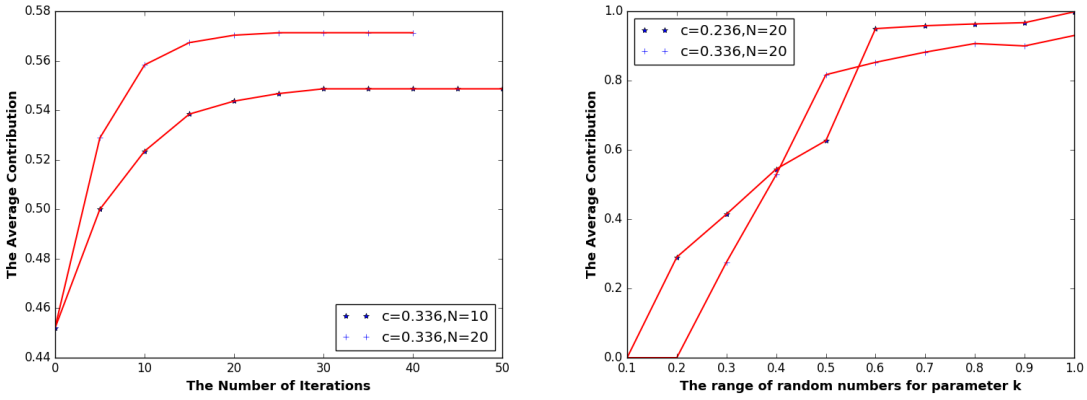
5.1 Synthetic Traffic Trace

We generate synthetic trace data based on the data set from MMSyS2015[1], which includes thousands of live streaming sessions of two major service providers: Twitch.tv and YouTube Live. From the data, we identify the most popular 50 videos based on number of views. Since each video includes log data regarding the number of viewers over time but the information regarding individual viewers is mostly missing (except for those who wrote comments), we simulate the behavior of each viewer in the following steps: (1) we divide time into time slots of duration 5 seconds; (2) at each time slot, each viewer would view content with probability proportional to the popularity of the content; and (3) based on the above content viewing procedure, an OTT can share content to other OTT devices if the content is already in its buffer. We set the buffer size at each OTT to hold content for 4 time slots. We simulated different number of viewers (i.e., 10 and 20).

Using the synthetic traffic trace, we illustrate how SmartSharing converges to equilibrium in the bandwidth game (Section 5.2) and how it reaches the desired

[1]Available from http://dash.ipv6.enstb.fr/dataset/live-sessions/.

pricing scheme (Section 5.3).



(a) (b)

Figure 5.1: Bandwidth game: (a) Average contribution against number of iterations to reach Nash equilibrium for 10 and 20 OTT devices. The average initial value of contribution per OTT is 0.4522713; parameter k_i is randomly chosen in $(0, 0.4)$. (b) Average contribution at Nash equilibrium against parameter k_i under different c for 20 OTT devices.

5.2 Bandwidth Game

Initially, we set OTT i's contribution to SmartSharing, $d_i (0 \leq d_i \leq 1)$, to a random value uniformly distributed in $[0, 1]$. Fig. 5.1 (a) shows average contribution at Nash equilibrium plotted against number of steps required to reach Nash equilibrium for 10 and 20 OTT devices. The average initial value of contribution per OTT is 0.4522713. The parameters c (i.e., unit bandwidth fee charged by the ISP) is set to 0.336. The result shows that with Algorithm BRUA, the average contribution increases with the number of iterations and quickly converge to equilibrium after 15 iterations. In addition, the average contribution for a system with 20 OTT devices is higher than that for a system with 10 OTT devices, implying that a larger number of OTT devices brings more chance of sharing and thus higher motivation for contribution.

To further investigate the impact of parameter c and k_i, we test SmartSharing with different settings of c and k_i. Note that k_i represents a weight of the content value to OTT i and c represents the bandwidth cost charged by ISP. Fig. 5.1 (b) shows the result. From the figure, the average contribution increases when the range of k_i changes from $(0, 0.1)$ to $(0.1, 1)$. This is easy to understand because a larger

range means a higher mean value of k_i over all OTT devices and thus a higher average utility value for OTT devices. In this case, OTT devices are willing to contribute more. Nevertheless, it is interesting to see that the average OTT contribution seems to have no clear correlation with the value of c. This implies that the resource that an OTT contributes to SmartSharing mainly depends on the value of its buffered content to other OTT devices rather than the bandwidth cost charged by the ISP.

5.3 Pricing Scheme

This section discloses the detailed intermediate steps on how OTT devices reach the pricing scheme with our proposed EM algorithm.

In this experiment, we show the pricing schemes of 10 OTT devices. The fetch matrix $\boldsymbol{X}$ was generated using the method introduced in Section 3.3. Since an OTT has the option to fetch content from the ISP directly or from other OTT devices, the prices charged by OTT devices must be cheaper than that charged by the ISP. Note that the price of ISP is set in parameter c. A price option for OTT devices could be denoted as γc where $\gamma \in (0, 1)$.

For ease of viewing, we first limit the price options to two prices: price 1 (high price) and price 2 (low price). Since customers tend to accept lower prices, the initial acceptance rates (i.e., the proportion of other OTT devices that would accept a price) to price 1 and price 2 are set to 0.2 and 0.5, respectively, i.e., $\boldsymbol{\theta}_0 = \{0.2, 0.5\}$. We then change the initial setting $\boldsymbol{\theta}_0 = \{0.3, 0.8\}$. Fig. 5.2 shows how $\boldsymbol{\theta}$ converges to stable values in our EM algorithm.

Comparing convergence results (Fig. 5.2) in bandwidth game with different initial $\boldsymbol{\theta}$ values, we conclude that the initial $\boldsymbol{\theta}$ values have no impact on their final converged values in our EM algorithm, and the converged results can be quickly reached after only several iterations. The converged results of $\boldsymbol{\theta}$ can help us find corresponding price values.

Table 5.1 shows the price strategies of 10 OTT devices with different fetch table $\boldsymbol{X}_i$. The result shows that the price strategy of an OTT is affected by its fetch table $\boldsymbol{X}_i$. OTT devices tend to choose a lower price if there are more $x_{ij} = 1$ (customers) in their fetch table $\boldsymbol{X_i}$. In other words, the price of an OTT reduces with the growth of the number of its customers.

We are interested in how the number of price options and the amount of content that an OTT shares with others impact the final price schemes. For this, we tested

Table 5.1: Price Strategies of 10 Different OTT devices

OTT	Percentage of $x_{ij} = 1$ in $\boldsymbol{X_i}$	$Q_{i,z(1)=0.46532c}$ (high price)	$Q_{i,z(2)=0.35884c}$ (low price)
1	44%	0.72136007129	0.27863992871
2	77%	0.254656968287	0.745343031713
3	77%	0.254656968287	0.745343031713
4	33%	0.835657536878	0.164342463122
5	77%	0.254656968287	0.745343031713
6	33%	0.835657536878	0.164342463122
7	55%	0.568606913951	0.431393086049
8	77%	0.254656968287	0.745343031713
9	55%	0.568606913951	0.431393086049
10	66%	0.401581763375	0.598418236625

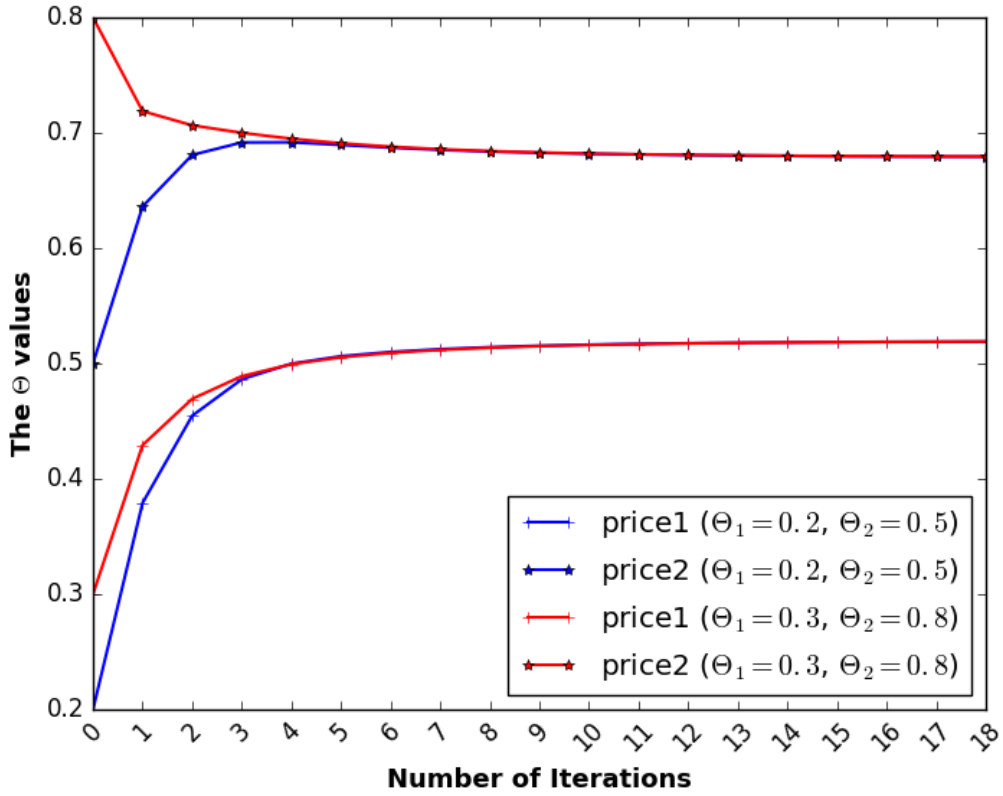


Figure 5.2: The initial probabilities that an OTT would fetch content from other OTT devices with price 1 and price 2 are set to 0.2 and 0.5, and then change to 0.3 and 0.8 respectively.

three cases: (1) two price options $\{0.3c, 0.8c\}$, (2) three price options $\{0.3c, 0.6c, 0.8c\}$, and (3) five price options $\{0.3c, 0.5c, 0.6c, 0.7c, 0.8c\}$. For ease of viewing, we only show the *average price* charged by each OTT. The average price is calculated by summing up the price value of each price option multiplied by the probability of this price option being adopted. The amount of content that an OTT shares with others is captured by the proportion of 1's in its fetch table. The higher proportion of 1's, the more content that this OTT can share with others.

Fig. 5.3 shows the average price versus the proportion of 1's in an OTT's fetch table, with different numbers of price options. From the figure, we can see that the number of price options offered to an OTT does not have clear impact on the average price charged by the OTT, because the three curves interweave with each other. Nevertheless, the average price drops with the increase of content amount that

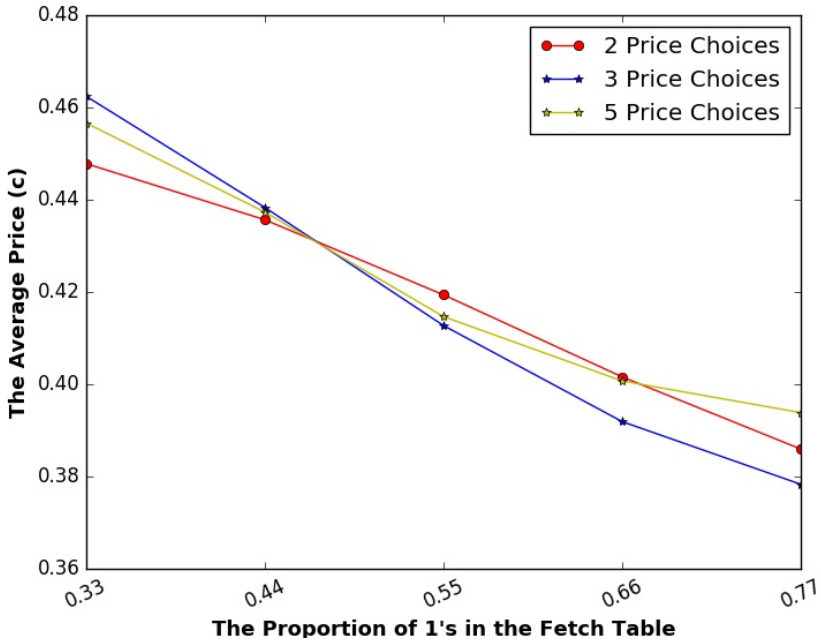


Figure 5.3: Average price vs. the proportion of 1's in the fetch table. 2 price choices: $\{0.8c, 0.3c\}$, 3 price choices: $\{0.8c, 0.6c, 0.3c\}$, 5 price choices: $\{0.8c, 0.7c, 0.6c, 0.5c, 0.3c\}$

an OTT can share with other OTT devices, since all the three curves show a clear down trend along the x-axis. This is reasonable because when an OTT shares more content with others, it can reduce the average price for the same level of profit.

5.4 Overhead of Smart Contracts

While the smart contracts only emulate the content transactions without involving actual content delivery, i.e., only meta data such as price and the content index is handled in the contracts, the smart contract-based evaluation over Ethereum sheds light on the speed and cost of executing transactions, which cannot be captured with trace-driven simulation.

We emulate the real-world transactions by deploying virtual machines (VMs) over two desktop computers connected by a local 10 Mbps Ethernet router. The two machines have same system configuration: Intel Core i7-7700 quad core 3.6 GHz CPU, 8 GB 2400 MHz DDR4 memory, and Window 10 Enterprise OS. Each VM represents

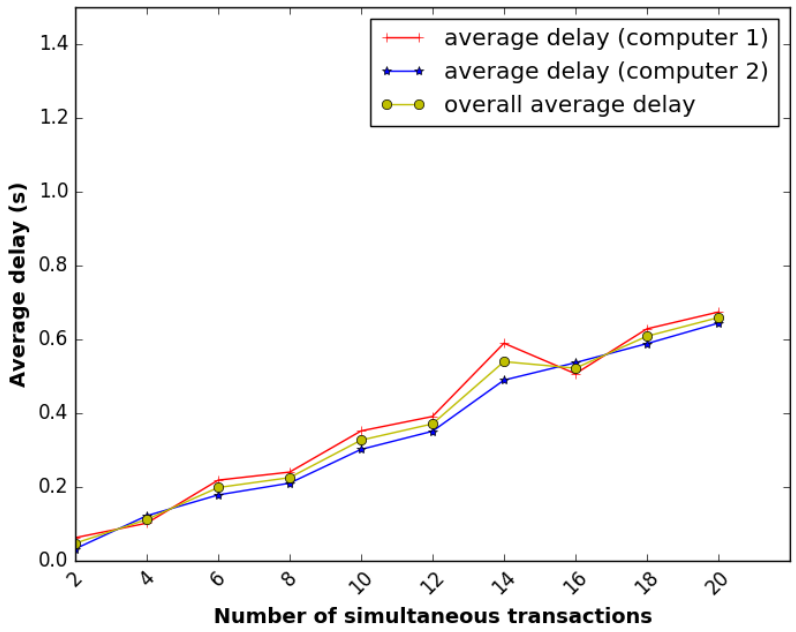


Figure 5.4: Average transaction delay (seconds) vs the number of simultaneous transactions

an OTT and runs smart contracts defined in SmartSharing. We test different total numbers of OTT devices, and for each test we deploy the same number of OTT devices in the two computers. We test the *worst-case scenarios* where transactions occur in burst. To be specific, we start a test by configuring several OTT devices to submit content transaction simultaneously, and end the test when all transactions have been executed. For each transaction, we record its delay which is calculated by its finish time minus its start time.

Fig. 5.4 shows the average delay of all transactions vs the number of simultaneous transactions. To give more details, the figure shows the average delay for OTT devices in each computer and the average delay for all OTT devices.

The average gas fee per transaction is about 0.00005 ether, which amounts to 0.03 USD according to the market price on the day of June 20, 2018.

The above results demonstrated that the overhead in terms of delay and monetary cost for executing smart contracts in SmartSharing is very small.

Chapter 6

Concluding Remarks and Future Work

6.1 Concluding Remarks

The idea of using end users' resource to boost CDN performance was proposed several years ago, and it has gained new momentum very recently due to the emerging blockchain technology [1]. Nevertheless, current blockchain-based CDN solutions are mostly proprietary with technical details unknown to the public. To clear the cloud, this thesis fully discloses the details of a new CDN solution, SmartSharing, which integrates a pricing scheme to incentivize end users and a smart contract-based content trading mechanism to facilitate content trading among end users.

Due to the special formulation of bandwidth game and the EM-based pricing, SmartSharing brings a win-win-win solution for CDN providers, end users, and content providers (CPs). It benefits the CDN providers by extending their cache footprint without deploying cache servers; it benefits end users by offering fast and cheaper content delivery from nearby OTT devices; it also potentially benefits CPs, since SmartSharing may boost more content views and more views imply higher profit for CPs.

Benefits to Content Providers

Video content is considered as the main stream of the Internet content market. Cisco has reported that Internet traffic used by video content will increase from 64% in 2014 to 84% in 2019. There are many popular Internet content providers, such as YouTube,

Netflix, and Tencent Video. It is challenging for content providers to distribute videos to billion of users. SmartSharing is of great value to develop a distribution content trading platform at low cost. The lower prices in SmartSharing can also help content providers attractive more viewers. In many cases, content providers earn profits by online advertisement embedded in video content. More viewers means potentially a higher conversion rate of advertisement and higher profit.

Benefits to CDN

Running a CDN system is expensive, and deploying a higher number of cache servers incurs higher equipment cost and heavier maintenance overhead for the CDN providers. With SmartSharing, ordinary users' devices can be used as mini-CDN servers. The resources and devices contributed by OTT owners in SmartSharing can effectively cut the CDN provider's expenditure on deploying and maintaining cache servers.

In addition, the rewarding scheme used to incentive OTT owners will not lead to the profit loss of the CDN provider, since the CDN provider does not need to issue any rewards to OTT owners. OTT owners compensate each other based on the content traded over SmartSharing.

Benefits to OTT users

With SmartSharing, since the bandwidth cost charged by an OTT device is set to be cheaper than the bandwidth cost charged by the ISP, obtaining content from nearby OTT devices is thus cheaper than obtaining content from the CDN server. As such, not only OTT users who sell the content to others can gain extra profit, but also OTT users who purchase the content from nearby OTTs obtain faster and cheaper services.

6.2 Future Work

6.2.1 Caching Strategies

We did not investigate different caching strategies in this thesis. We simply assumed that a user download content that she/he will be viewing and share it to other nearby users if needed. Equivalently, users in SmartSharing would view content with probability proportional to the popularity of the content.

To further improve SmartSharing, it is important to investigate different caching strategies. Previous studies of caching strategies can be divided into two categories. The first, global scheduling solutions, determine the content placement based on global content popularity prediction and ignore differences among user interests in different locations (the solution we have used in this thesis). The second category mainly explores mathematical analysis for content placement problems by maximizing the request ratio that could be served to particular users. For instance, Sun [17] used the content title to study users' affinity to the contents and their video transition behavior. They find that users tend to continue to watch the current TV episode or the next three TV episodes, with the probability of 35% and 40%. They model customers' transition behavior among TV series as a Markov decision problem based on this sight. Reinforcement learning algorithm was adopted by Sun [17] to design strategies to solve the challenging problems of content popularity prediction.

6.2.2 Clustering OTT devices

Another problem needed to be discussed in the future is the partition for OTT devices. In SmartSharing, OTT devices are grouped by different local areas, and only OTT devices in the same area can serve each other.

Another strategy is to group users (thus OTT devices) with the similar interests, since in this way the chance of content sharing in the same group will become higher. Actually, we can calculate users' interest similarity index based on the Jaccard similarity coefficient and then divided OTT owners into subgroups based on their similarity index. Users in different groups will have different content preferences.

Nevertheless, this method may also bring new problems if geographically distant OTT devices are grouped together. The delivery delay and the bandwidth cost will become much more complicated. Studying different design options in such a system will be an interesting research problem.

Bibliography

[1] A Distributed CDN platform Based on Blockchain Technology. `https://www.blockcdn.org`.

[2] Alex Buonassisi and Jennifer Marles. Can you stream your way around copyright infringement in canada? `https://patentable.com/can-stream-way-around-copyright-infringement-canada/`, accessed in June 2018.

[3] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. A game theoretic framework for incentives in p2p systems. In *Peer-to-Peer Computing, 2003.(P2P 2003). Proceedings. Third International Conference on*, pages 48–56. IEEE, 2003.

[4] Jesse Chandler, Gabriele Paolacci, and Pam Mueller. Risks and rewards of crowdsourcing marketplaces. In *Handbook of human computation*, pages 377–392. Springer, 2013.

[5] Xu Cheng and Jiangchuan Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *Infocom 2009, Ieee*, pages 1152–1160. IEEE, 2009.

[6] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[7] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[9] Syed Hasan, Sergey Gorinsky, Constantine Dovrolis, and Ramesh K Sitaraman. Trade-offs in optimizing the cache deployments of cdns. In *INFOCOM, 2014 Proceedings IEEE*, pages 460–468. IEEE, 2014.

[10] Nicolas Herbaut and Nicolas Negru. A model for collaborative blockchain-based video delivery relying on advanced network services chains. *IEEE Communications Magazine*, 55(9):70–76, 2017.

[11] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 839–858. IEEE, 2016.

[12] David G Luenberger. Complete stability of noncooperative games. *Journal of Optimization Theory and Applications*, 25(4):485–505, 1978.

[13] Qian Ma, Lin Gao, Ya-Feng Liu, and Jianwei Huang. Economic analysis of crowdsourced wireless community networks. *IEEE Transactions on Mobile Computing*, 16(7):1856–1869, 2017.

[14] Stephen Pryke. Game theory and networks. *Managing Networks in Project-Based Organisations*, pages 77–94.

[15] D Rayburn. The state of the cdn market: Video pricing, contract, volume and market sizing trends, 2015.

[16] David Roger Smart. *Fixed point theorems*, volume 66. CUP Archive, 1980.

[17] Lifeng Sun, Ming Ma, Wen Hu, Haitian Pang, and Zhi Wang. Beyond 1 million nodes: A crowdsourced video content delivery network. *IEEE MultiMedia*, (3):54–63, 2017.

[18] Petar M Vasić and Josip E Pečarić. On the jensen inequality. *Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika*, pages 50–54, 1979.

[19] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[20] Yipeng Zhou, Terence H Chan, Siu Wai Ho, Guoqiao Ye, and Di Wu. Replicating coded content in crowdsourcing-based cdn systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[21] Yipeng Zhou, Liang Chen, Mi Jing, Shenglong Zou, and Richard Tianbai Ma. Design, implementation, and measurement of a crowdsourcing-based content distribution platform. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(5s):80, 2016.