November 2014

Design and Implementation of a Network Service Marketplace

Yunsheng Qi
University of Massachusetts Amherst

DESIGN AND IMPLEMENTATION OF A NETWORK SERVICE MARKETPLACE

A Thesis Presented

by

YUNSHENG QI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2014

Electrical and Computer Engineering

DESIGN AND IMPLEMENTATION OF A NETWORK SERVICE MARKETPLACE

A Thesis Presented

by

YUNSHENG QI

Approved as to style and content by:

Tilman Wolf, Chair

Michael Zink, Member

Weibo Gong, Member

C. V. Hollot, Department Head
Electrical and Computer Engineering

To my beloved parents.

Many thanks to Professor Tilman Wolf.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my advisor Prof. Tilman Wolf for his invaluable guidance during the project and his instruction on my work.

Then I would like to thank PhD student Mr Xinming Chen, who offered plenty of zealous help and his suggestions that helped me to reach the final design of the probe.

I will also give my gratitude to PhD student Mr Abhishek Dwaraki. Without his help of database design and service specification, I know I cannot reach next stage of the project.

In addition, the PhD students Mr Jiahui Jin, Ms Samamon Khemmarat and Mr Guoyi Zhao provide me a lot of useful advice on my project.

ABSTRACT

DESIGN AND IMPLEMENTATION OF A NETWORK SERVICE MARKETPLACE

SEPTEMBER 2014

YUNSHENG QI

B.Eng., TIANJIN UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Tilman Wolf

The Internet has successfully served the world for more than three decades, while limitations and drawbacks still exist. A lot of researches have been done on innovation of Internet to address those inadequacies. One approach to improve Internet performance is to make *choice* as a principle in network architecture. We believe that a service-based network architecture with choice, or we call it *ChoiceNet*, is the most suitable option for future Internet. In this thesis, we design and implement a network service marketplace. Due to the flexibility and functionality of web application, the marketplace is implemented by a web application to fulfill functions such as storing and searching network services, handling actual financial transaction and interacting with client and service planner.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Background

The Internet has amazingly developed in both technology and theory in past 30 years. According to the statistics of Internet World Stats, the growth of world internet users is 566.4% from 2000 to 2012 [8]. With the extreme growth, the current Internet has successfully supported a wide range of services and applications at the edge of the network [12]. In spite of the success of Internet, there are still a lot of limitations and inadequacies in current network architecture with emerging requirements for the Internet including security, privacy, personalized service, complexity of management etc. Therefore, new network architectures are being explored.

There are several research teams currently studying next-generation Internet. Some of them focus on data transmission, such as Named Data Networking (communication driven by the receiving end [18]) and some focus on cloud computing, such as NEBULA Future Internet Architecture (a future network that enables the vision of cloud computing to be realized [1]).

In this thesis, the research focuses on service-based network with *choice*. "Design for choice" is a key principle for future Internet [4]. We believe that *ChoiceNet* (service-based network with choice) is the most suitable option for future Internet since it can not only drive innovations of current Internet but also take economy plane into account as a crucial place. According to [16], choice means that

> ... the entities using the network can select from a range of alternative services that may differ in functionality, performance, and cost ...

Choice can allow the dynamic selection of those alternative services if the network architecture supports it. It is also necessary to ensure that incentives trigger innovation through suitable economic process and users can "vote with their wallets" [16].

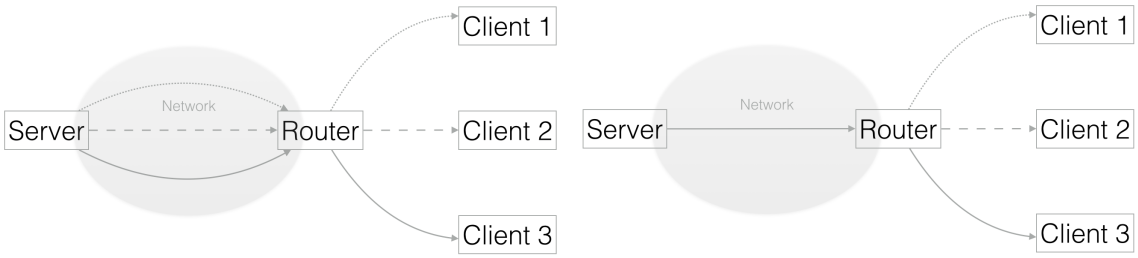
1.2 Motivation

Research institutions and business companies never stop to explore the possibilities to improve the performance of Internet through innovation network architecture since the invention of Internet. Some of them are success and deployed to commercial use, while others are just to stay in the research stage and have not or slowly been used in real life for a variety of reasons. A famous one of them is *IP Multicast* which is lacking consideration of commercial world.

IP multicast is a method of sending Internet Protocol (IP) datagrams to a group of interested receivers in a single transmission [15]. The sender only sends the packet once and the nodes (switches and routers) take care of delivering packet to multiple receivers by a multicast distribution tree structure. Figure 1.1 shows the comparison of unicast, which we are currently using, and multicast. There is no restriction for hosts or users to create a multicast group, sending data to a group or receiving data from a group and receivers do not have any kind of access control [5].

As with all IP datagrams, multicast datagrams are best-effort. The efficiency advantage of IP multicast is apparently. However, there are also some existing issues, such as group management, distributed multicast address allocation and security and privacy [5]. Those issues are significantly increasing the cost of transit Internet service providers (ISPs) and reducing data traffic which is source of their profits. Without additional profits and growth of expenses, the multicast has been under slow commercial deployment by ISPs and carriers. ChoiceNet can provide solutions which take economy plane into account to address the commercial deployment of IP multicast or other network innovations. For example, users have choice to switch to

multicast or stay on unicast, while service providers can charge more for multicast to balance high expenses on multicast.



(a) Unicast. (b) Multicast.

Figure 1.1: Unicast v.s. Multicast.

Due to the importance of economy plane in ChoiceNet, a medium is necessary between the user and service provider. (In this thesis, user is defined as any entity using service and service provider is the entity can provide choice of service. Therefore, access ISP can be a user to use the service from transit ISP and also can be the service provider to provide network service to customers.) A web application is implemented to act as this medium because of its advantages of multiple functions, flexibility on multiple platforms and easiness on extending functions.

1.3 Problem Statement

There are a lot of problems to design and implement a web application to meet the requirements of ChoiceNet. This section states those problems and solutions will be proposed in next few chapters.

The first problem before design the ChoiceNet web application is the definition and description of service. Each network service is a choice or one network service provides multiple choices. In ChoiceNet, one entity could act as multiple roles. For instance, Comcast, an ISP, buys services from AT&T, a backbone network operator,

and also provides services to their customers. In that scenario, services are varied and complicated. A service could be an independent service or a composition of a series of services. It is hard to figure out a general way to define and describe a service to meet different conditions. Definition and description of service are also prerequisites for data storage and service providing in the web application.

After explored the solution of service definition and description, problems are appeared for designing the web application. The following concerns are requirements for this web application, but not limited on those.

1. Service provider can submit service to this web application
2. The web application have storage for those services
3. User can search and choose service easily
4. The web application can handle financial transaction between user and service provider
5. The web application can set the service up once service provider received user payment
6. Service provider can manage his account and proposed services
7. User can manage his account and purchased services
8. The web application should have friendly interface for both user and service provider
9. The communication between server and client app should be secured

There are much more details on those requirements, such as how to make the search and display services fast, how to handle multiple requests for one service and how to protect user privacy and information. The rests concerns will be discussed in next two chapters.

1.4 Contribution

This thesis is centered around the marketplace implementation of ChoiceNet, aiming to build up a web application and deploy multiple functions. First, we review related researches and works which can solve some initial concerns such as the description of service and economic benefits of ChoiceNet. Next, the design of web application is proposed to solve every problem in detail, including database table design and optimization, using PayPal to handle financial transaction and combination of server and client method to process users' searching service request efficiently. Finally, the results and discussion are listed.

The main contributions are as below:

1. Implement the data storage for service according to the service definition and description.

2. Design the searching algorithm and user interface for user searching and choosing service.

3. Handle actual financial transaction between user and service provider.

4. Hand over the network service payment information to planner who can set up network and provide service to users.

5. Establish the secured data transmission between server and client

1.5 Organization

The remainder of the thesis is organized as follows: Chapter 2 presents the background and related work on ChoiceNet. This chapter is the basis of further research of ChoiceNet web application including the definition of choice, ChoiceNet architecture, specification of service, economy plane and cryptography. Chapter 3 introduces the design of web application in detail. This chapter discusses the solutions of previous

concerns such as database design, web interface and financial transaction. Chapter 4 provides results of web application and discussion of them. In Chapter 5, we present the conclusion and further work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we introduce related researches and works which are the foundations of future steps. We first present the answer of question - "what is ChoiceNet?" and explain the crucial role of choice. Next section gives a brief overview of the architecture of ChoiceNet. Section 3 introduces the specification and composition of network services, which clear the initial concern of service definition and description. Section 4 presents economic concerns including economic plane in ChoiceNet and possible solution of financial transaction. Last section shows some theories of cryptography related to our web application.

2.1 Service-Based Network with Choice

2.1.1 What is ChoiceNet?

In previous section, we define choice as the network users have the opportunity to select a service from a range of alternatives which are different in functionality, performance, and cost. Choices can and should appear at different layers in the protocol stack of a network, ranging from different communication paths to different protocols and application-layer services [12]. It is also necessary to ensure that incentives trigger innovation through suitable economic process and users can "vote with their wallets" [16]. There are a large amount of studies that research various economic issues effect current network and Internet, but most of them trend to analyze and understand the effects. ChoiceNet system integrates the economic processes and interactions into the network architecture [12]. Through this work, researchers can

concentrate on innovations of network and leave the commercial deployment to the marketplace of ChoiceNet.

2.1.2 Choice as Principle in Network

ChoiceNet system is based on following three key principles [16]:

Principle 1: Encourage alternatives to allow users to choose among a range of services. The underlying network architecture must provide the support to create different types of services and to create alternative services of the same type [16]. Alternatives allow users to select the best choice for their applications, ranging from value for money to best performance to meet their needs. For example, in present Internet, users have to switch to other ISPs if they are not satisfied with service provider. In ChoiceNet, users can select more specific services, such as specific route to transmit their data or RSA to encrypt their data.

Principle 2: Let users vote with their wallets to reward superior and innovative services. In current network or Internet, the competition only exists at the application layer, while ChoiceNet allow choices appear at different layers in the protocol stack of a network. The underlying network infrastructure has the mechanism that users encourage service provider to offer superior services and discourage inferior or outdated services with users' wallet. In ChoiceNet, providers with more innovation and high quality service have more chance to survive in competition. For instance, users trend to pay the service with best value for money and service providers have to deploy more advanced technology and develop more service innovation.

Principle 3: Provide the mechanisms to stay informed on available alternatives and their performance. ChoiceNet should have the mechanism to keep both users and providers updated from market information. That means users can find services which meet their needs easily and be informed about new services which are better than those they are currently using. At the same time, service provider

can acquire information from user to update their services to better match market demands. This incentive of medium between users and providers makes ChoiceNet web application more significant.

These three principles are interdependent and mutually rely on each other [16], which are illustrated in figure 2.1. For instance, users will not pay for innovative services if they don't know the existence of innovations. Users' "vote with wallet" will stimulate providers to update or innovate services and alternatives are the information for users to "know what happened".

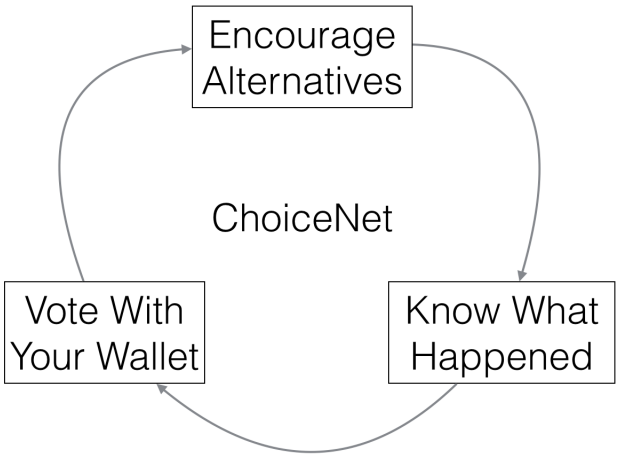


Figure 2.1: Foundational principles and their dependencies [16].

2.2 ChoiceNet Architecture

The three principles of ChoiceNet arouses an entirely new network architecture which has choices inside the network. Figure 2.2 illustrates schematically how the principles interact within ChoiceNet and the new features they support.

Based on the first principle, "encourage alternatives", of ChoiceNet, a new control-plane and economy-plane mechanisms are offered to support alternatives. The marketplace in economy-plane allow service providers to publish and advertise their services, while users or applications can select alternative services through negotiation

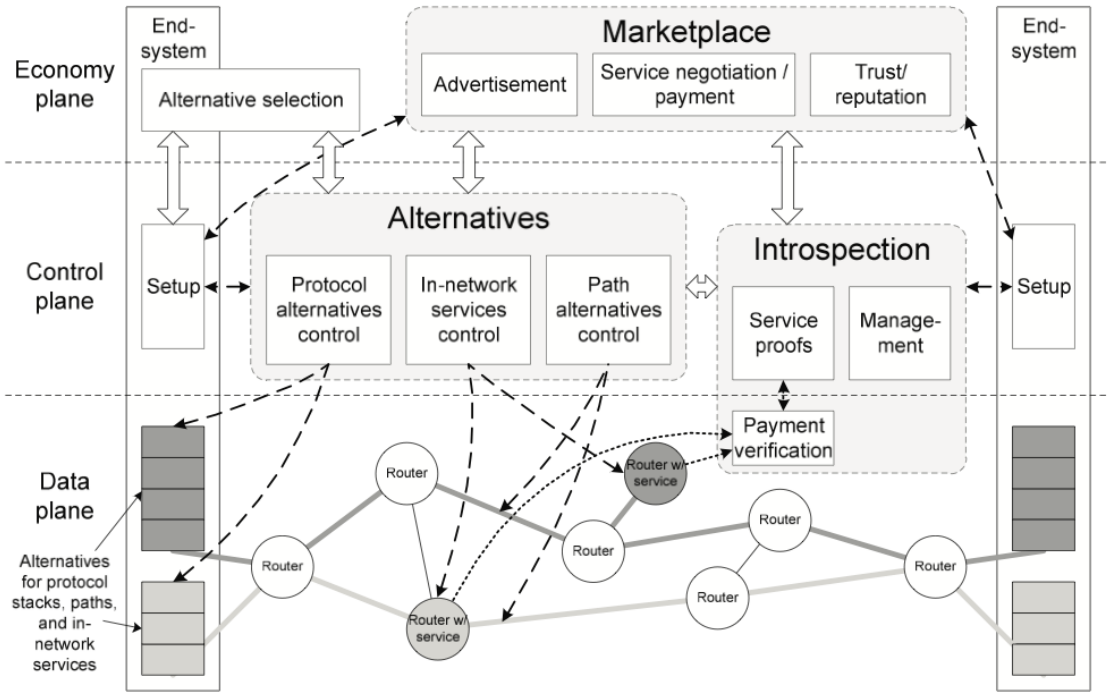


Figure 2.2: Service composition framework in ChoiceNet [16].

and payment. Users can "vote with wallet" to reward innovations. The selected service is set up as an optimal path based on users' demands in control-plane and ChoiceNet also has introspection mechanism to keep providers and users informed according to the principle, "knowing what happened". Data-plane can be current or innovative network infrastructures to implement the selected service, which is called service composition. Next section shows how varied services to be described and composed in a standard way. Economy-plane is entirely new in network architecture so we will presents more details in section 4.

2.3 Specification and Composition of Network Services

Based on the work in [3] and [13], one of important prerequisites to design this web application is solved. This section introduces the description and composition of service in ChoiceNet.

2.3.1 Service Description

Due to variety of services, it is hard to generalize a description of service. Based on the definition of choice and characteristic of ChoiceNet, service should include everything in the network, such as link bandwidth, storage or software services offered on nodes [3]. A service can be provided as multiple services by one service provider and third-party entities can also offer aggregated service by multiple providers. This means more complexity of service. A description of service should let providers submit services to the web application in a standardized manner. Therefore, we describe a service as:

$$S =< IFset, \Delta_{pre}, \Delta_{post} > \tag{2.1}$$

where $IFset$ is the service's interface set that defines all the I/O interfaces; Δ_{pre} are *preconditions* that need to be satisfied for the service to be executed; and Δ_{post} are *postconditions* of the output produced by the service. Both these sets of characteristics include protocol and data semantics [6].

In most of the scenarios of using this ChoiceNet, users are tending to use a combination of multiple services. Therefore, service composition and instantiation will present a solution to chain multiple services.

2.3.2 Service Composition and Instantiation

Service providers and developers can submit services to web application based on this standard. The web application can than store those services and display them to users. When users select a service and pay for it, the marketplace has to response to it and provide service. To be more specific, the marketplace has to compose a set of services along a path which are optimization or near optimization in cost, latency or bandwidth with respect to the users' selection, and marketplace inherently dictates the path to be taken and the order in which services are executed [3].

Service composition is the process of putting a series of basic services together to form a complicated service which can be offered to customers [6]. As service description mentioned in last section, a service can be represented as a sequence of services (S_{i1}, $S_{i2}...S_{in}$) and qualified with $< I, O >$, where I is the precondition of S_{i1} and O is the postcondition of S_{in} [6].

Service instantiation happens after customer chooses a service. Marketplace hands service information to planner who can find proper path. Once appropriate paths have been identified, the planner hands them back to marketplace to make final choice.

Figure 2.3 shows examples of service composition. C_1 is a video request from end system $ES1$ to video server $Server2$. $R2$ is required for format conversion, because the formats of video sent from $Server2$ and displayed in $ES1$ are not same. Different formats consists of part of preconditions and postconditions in this service. Moreover, high bandwidth is also necessary for video transmission in this service. Therefore, the composition of C_1 is:

$$S =< Server2, E1, R1, R2, R4, E2, ES1 > \tag{2.2}$$

Similar to C_1, C_2 is a low bandwidth video request and its composition is:

$$S =< Server2, E1, R2, R3, E2, ES1 > \tag{2.3}$$

C_3 is a request of financial data transaction with encryption over a low bandwidth link. The service composition for C_3 is:

$$S =< Server1, E1, R2, R3, E2, ES1 > \tag{2.4}$$

This problem can be reduced to an Artificial Intelligence (AI) planning problem. We can represent every communication request as:

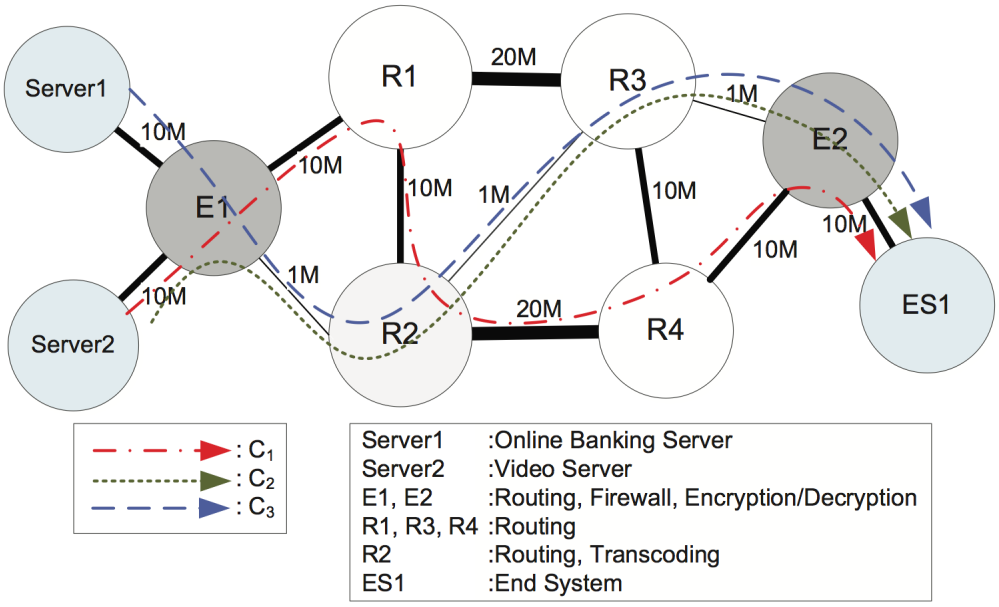


Figure 2.3: Service composition framework in ChoiceNet [3].

$$S =< I, G > \tag{2.5}$$

where I is the *precondition* of the source service, and G is the *precondition* of the destination service [3]. This will be able to help us obtain optimal service compositions.

2.4 Economy in ChoiceNet

2.4.1 Economy Plane

In current network or Internet, customers pay access ISPs for Internet access and service, while access ISPs pay transit ISPs and other ISPs for carrying their traffic to/from the rest of network [17]. A simple transmission across the Internet always include at least three different network providers - access ISPs to transit ISPs to another access ISPs. In this economic relationship, transit provider have no incentive to compete for customers by offering service and users have no control on the quality of received services. Current economic relationship lacks a rational mechanism for users to reward their network provider for quality and innovation network services.

ChoiceNet economy plane aims to allow network service providers compete for users and users can vote providers by their wallet for better services.

The current Internet is a complicated system which is beyond most people imagination and no single network provider can control all end-to-end path [17]. ChoiceNet enables service providers to offer single network service which they are operates, and ChoiceNet also allows virtual providers or third parties to provide services which are combination from others. We discuss an example of online video chatting to explain how economy plane to implement ChoiceNet idea. Online video chatting providers, like Skype on PC and Line on mobile platform, can only optimize their service by improving terminal and compressing data. Those providers almost have no control of data transmitting on physical infrastructure and users are not able to choose which network provider to handle their video chatting data. ChoiceNet enables online video chatting providers to provide services with integrating network services from ISPs or third party entities to offer services by combining services from Skype and network providers.

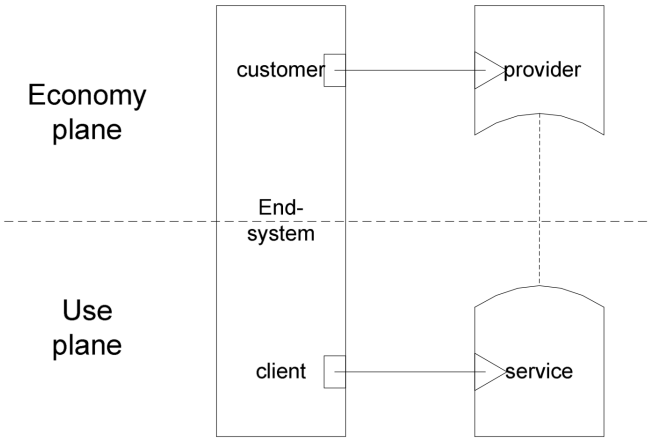


Figure 2.4: Interfaces in ChoiceNet [12].

Figure 2.4 illustrates the relationship of interfaces in economy plane and use plane (use plane is the traditional control and data plane in figure 2.2). In economy plane,

customers and providers interact with each other to establish economic relationships in which customers can choose and pay for services providers offered. In use plane, services are enabled based on economy plane agreements. ChoiceNet comprises economy plane and use plane [17].

[10] and [9] develops a game theory model for ChoiceNet in which network service providers compete with the quantities of services and maximize their profit.

2.4.2 Financial Transaction

Financial transaction in real world is not as simple as users making payment and got services. Financial transaction includes the identities of users and service providers, the trust between users and service providers and the security of financial transaction. Each user and service provider must have unique ID for users to distinguish providers and for providers to identify users to provide services. Trust is another important factor. ChoiceNet should have a trust mechanism to protect users and service providers from malicious entities. Moreover, financial security is the concern of every entity.

Based on those requirements, PayPal is selected to realize financial transaction with real financial world. PayPal is not only meet ChoiceNet needs, but also a third party e-commerce business which allow users to have multiple choices to pay through Internet.

2.5 Cryptography and Authentication

Due to the complicated characteristic of financial transaction in real world, it is very important to take some steps to protect it. This section introduces some encryption methods which are used in the web application and the secured method to exchange key between server and client.

2.5.1 Secure Hash Function

The purpose of hash function is to generate a "fingerprint" of a file, a message and other block of data [14]. Hash function has a lot of properties, such as any size input data, fixed size output data and easy implementation, however properties of "one way computation" and "collision resistant" make secure hash function to be the best method to be used in user authentication. Figure 2.5 shows how hash function works in user authentication. The password is transmitted and stored as hashed-password rather than plain text. Even the transmitting path or server database is compromised, it is impossible for attacker to obtain user password because of "one way computation" which means it is beyond the ability of malicious agent to beak it with existing resources.

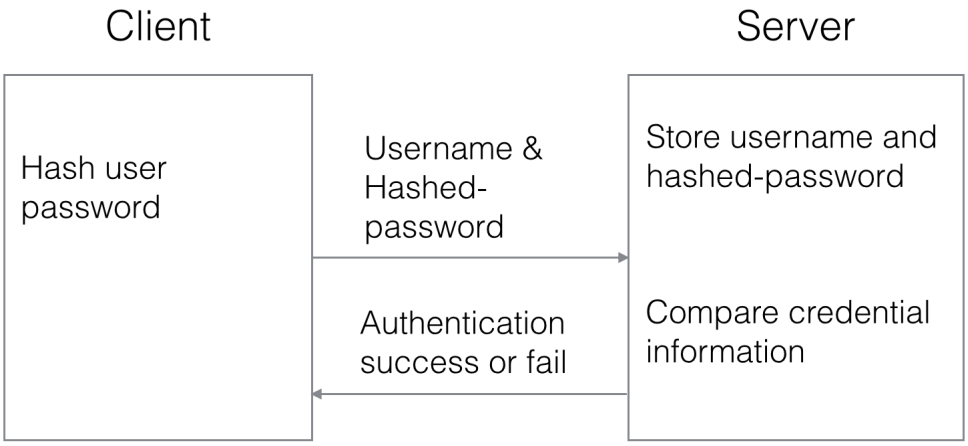


Figure 2.5: User Authentication.

2.5.2 Symmetric Encryption

The communication between server and client is required to be encrypted to protect payment information and other sensitive information. The plain text can be encrypted and cipher text can be deciphered by both server and client with same key. Therefore, key exchange become a crucial part and this is discussed in next section.

Common symmetric encryption methods are Data Encryption Algorithm (DES) and Advanced Data Algorithm (AES).

2.5.3 The Diffie-Hellman Key Exchange Algorithm

The purpose of Diffie-Hellman (DH) key exchange algorithm is to let two entities exchange key securely and then they can use it in following encrypted communication. The following steps summarizes the Diffie-Hellman Key Exchange Algorithm.

- **Global Public Elements:** Both server and client obtain the same q (prime number) and a ($a < q$ and a is primitive root of q).
- **Public Key Generation:** Server and client select X_s ($X_s < q$) and X_c ($X_c < q$) and calculate public key Y_s ($Y_s = a^{X_s} \bmod q$) and Y_c ($Y_c = a^{X_c} \bmod q$) respectively. Then, server and client exchange Y_s and Y_c with each other.
- **Secret Key Generation:** Server generates secret key K by $K = Y_c^{X_s} \bmod q$ and client generates same secret key K by $K = Y_s^{X_c} \bmod q$.

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponential modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible. [14]

CHAPTER 3

DESIGN OF THE WEB APPLICATION

This chapter presents the design of ChoiceNet web application and we discuss solutions for each concern in previous statement. In this chapter, we first give an overview of the whole implementation of ChoiceNet and the design of web application. The design of database is introduced in section 2 and each model is depicted in detail in subsection. The following sections show the design of web interaction and interface, financial transaction, searching and filtering and client app interaction.

3.1 Overview

3.1.1 ChoiceNet Implementation Overview

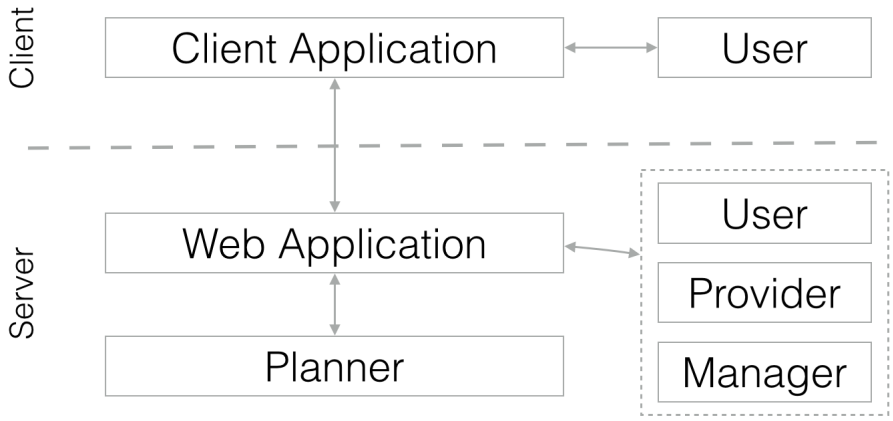


Figure 3.1: ChoiceNet implementation overview.

The architecture of implementation is shown in 3.1. The implementation of ChoiceNet contains three main parts which are client application, web application

and network planner. At client side, we have a client application which can detect any new connection in client machine and provide corresponding services to user. At server side, the web application handles service management, account management and financial transaction. Network planner calculates candidate paths for user selected service and provide corresponding service.

3.1.2 Web Application Overview

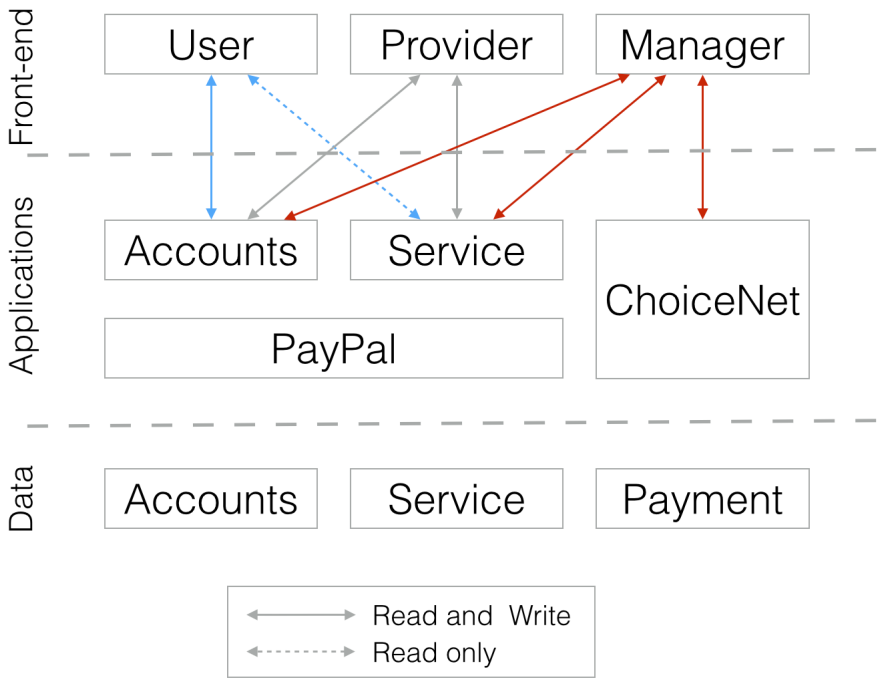


Figure 3.2: Web application overview.

Figure 3.2 shows the overview of the whole web application. The main purpose of this web application is to implement the economy-plane of ChoiceNet. Essential functions of marketplace are: users and providers can exchange information which can help users to choose right service to meet their needs and help providers to find market demands and offer high quality and innovative services. Moreover, it can handle payment between users and providers and financial transactions with real world. It then hands over the service information to planner to set up corresponding network and

provide service to users. Therefore, the web application is more like an e-commerce web application selling services and locates the application layer of ChoiceNet. This web application is equipped with multiple functions to solve problems of ChoiceNet.

3.1.3 MVC Design Pattern

Model-view-controller (MVC) is a popular software design pattern for implementing user interfaces. It divided a software application into three interconnected parts: *model*, *view* and *controller*. The *view* manage graphical and/or textual outputs which are displayed to users; the *controller* interprets inputs from the users; finally, the *model* manages the behavior and data of the application and interact with *view* and *controller* [2]. Figure 3.3 is a typical relationship between MVC and users.

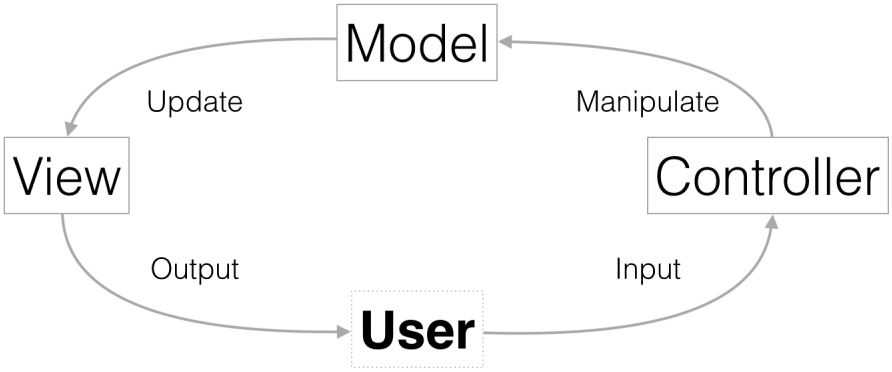


Figure 3.3: MVC components.

In web application, *model* is the "manager" of database to crunch data. *View* retrieves data from *model* and displays them to users in web page. *Controller* processes the user input, such as username and password, to *model*. Django is a web framework based on the idea of MVC. Because Django can build deep, dynamic, interesting sites in an extremely short time [7], we choose Django as our web application framework.

3.1.4 Sub-Applications

The web application should be divided into parts, which are *sub-applications*. Each of them aims to solve one problem or some problems and cooperates with each other as a whole. There are several benefits to use sub-applications. First, it is easy to update the web application. Painful searching and finding in whole web application to do little modification become eliminated and new complicated function can be added as a new sub-application. Second, work can be distributed by sub-application. Though most of the web application part is going to be done by one person, it is part of a big project. Group work is necessary in some parts like handling service routing calculation and service delivery. Last, the web application can have a clear outline for others to read and use.

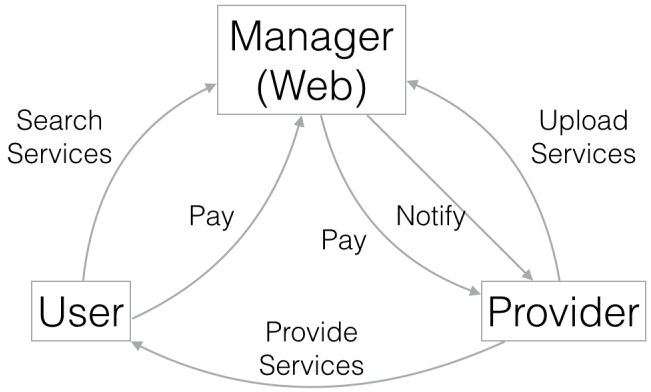


Figure 3.4: Accounts relationship.

There are four sub-applications in ChoiceNet Shopper:

- **accounts:** It is the sub-application to handle user accounts including user signing up and login, user information storing and changing, etc. Accounts can be categorized to three types: *manager* who is in charge of the web application, *provider* who provide and advertise services, *user* who utilizes and pays for services. Figure 3.4 shows the relationship of three types of account.

Table 3.1: Requirements and corresponding sub-applications.

Requirements	**Corresponding Sub-applications**
Service provider can submit service to this web application	accounts, service
The web application have storage for those services	service
User can search and choose service easily	accounts, service, choiceNet
The web application can handle financial transaction between user and service provider	paypal, choiceNet
The web application can set the service up once service provider received user payment	paypal, choiceNet
Service provider can manage his account and proposed services	accounts
User can manage his account and purchased services	accounts
The web application should have friendly interface for both user and service provider	accounts, service, choiceNet
The communication between server and client app should be secured	accounts, choiceNet

- **service:** This sub-application is the essential part of this web application and manage all of services information. It can retrieve data from database and display it based on user requirements. User can choose services and get served after paying for them. The services in this sub-application follow the requirements of service specification and composition in previous chapter.

- **paypal:** It is the sub-application to handle actual financial transaction and store payment data. This sub-application is a standard application for PayPal and integrates with PayPal IPN (Instant Payment Notification).

- **choiceNet:** ChoiceNet is a comprehensive sub-application. It handles user balance, provider income, service comment, service invoice and interaction session with client app.

Each of the sub-application aims to handle part of the stated requirements of this web application. For example, when user search a service, web application search services in service sub-application based on keywords user submitted; then filter the results based on user permissions from accounts sub-application; finally, display results to users by choiceNet sub-application. Table 3.1 shows those requirements and corresponding sub-applications.

3.2 Data Storage Design

3.2.1 Data Storage Overview

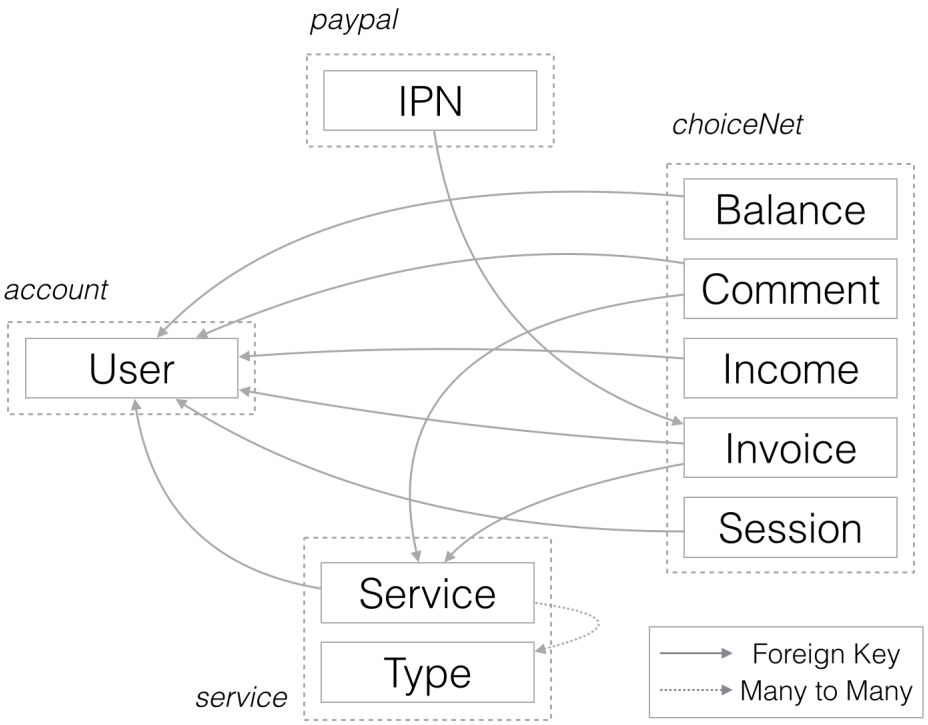


Figure 3.5: Database relationship.

Database design is the essential part of this project. The database design is the dominating factor of the operation efficiency of the whole web application. Designing too many tables means more joining times when querying the database. Joining table is very inefficiency for server operation. Too few tables lead to one or two big table, which also takes time to search expected items. One of the rules to design database is to keep frequently used attributes in main table and use foreign-key or many-to-many-field to link not common used attributes in other tables. Figure 3.5 shows the overview of the relationship of tables.

Each sub-application has its own tables for its function. We presents paypal sub-application as a section and next three subsections introduce the design tables of accounts, service and invoice respectively.

3.2.2 Accounts Sub-application

Table 3.2: Users table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
username	VARCHAR	70	Use email as username
password	VARCHAR	128	Encrypted by SHA-256
first_name	VARCHAR	40	-
last_name	VARCHAR	40	-
isSuper	TINYINT	1	Boolean, web super user has more priority
accountType	VARCHAR	255	Classify users
is_active	TINYINT	1	Only active user can use this web
is_staff	TINYINT	1	Staff user can manage services
date_joined	DATETIME	-	The date and time user signed up
last_login	DATETIME	-	Auto added when user login
is_superuser	TINYINT	1	Super user can login admin page

Accounts sub-application has one table: users. Table 3.2 lists the details of users table. The password is stored in encryption by SHA-256 to protect user information. Super user is the manager of the web application including database, user and provider behaviors. Account type classifies users into three types which are *user*, *provider* and *manager* respectively. *User* is the customer of network service. They pay for selected services and get the corresponding services. *Provider* can be physical network infrastructure provider or third party provider. They offer single service or complex services, which are combination of related services, to users. *Manager* has the highest priority of this web site. They can manage all of the services provider uploaded and all of the accounts of user and provider.

3.2.3 Service Sub-application

Table 3.3: Service table.

Name	**Type**	**Length***	**Description**
id	INT	11	Auto added, auto increment
name	VARCHAR	20	-
picture	VARCHAR	100	Path of picture
owner	FOREIGN-KEY	-	To accounts user table
description	VARCHAR	3000	-
service_id	VARCHAR	255	Public service ID
service_type	MANY-TO-MANY	-	To servicetype table
service_cost	DECIMAL	64, 12	-
service_bandwidth	DECIMAL	64, 12	-
service_lantency	DECIMAL	64, 12	-
** (20, 2) means the length of the number is 20 with 2 decimal places.*			

Service application contains two tables: *service*, *servicetype*. The important details of service table is listed in table 3.3. This table stores services, which are defined and described in previous chapter. The most frequently use attributes are list in this table, while others related to other table.

Table 3.4: Servicetype table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
name	VARCHAR	20	-
category	VARCHAR	20	Examples: “important”, “not important”
description	VARCHAR	200	To accounts user table

Table 3.4 shows servicetype table. Service types are what kind of services provided to users, such as encryption, decryption, compression, decompression, coding, DNS, WINS, link, etc. Node and link table are similar to servicetype and not listed here.

3.2.4 ChoiceNet Sub-application

Because choiceNet sub-application is comprehensive and handles complicated functions, it has five tables: balance, comment, income, invoice and session.

Table 3.5: Balance table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
user	FOREIGN-KEY	-	To accounts users table
balance	DECIMAL	64, 12	-

Table 3.5 shows the details of user balance. This table store the current balance of user.

Table 3.6: Comment table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
user	FOREIGN-KEY	-	To accounts users table
service	FOREIGN-KEY	-	To services service table
rate	INT	11	0-5 rate
comment	VARCHAR	4096	The comment left by user or provider
created_date	DATETIME	-	-
is_provider	TINYINT	1	Check commentator is provider or not

Table 3.6 shows the details of comments. This table store rating from users and comments left by user or provider to service.

Table 3.7: Income table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
provider	FOREIGN-KEY	-	To accounts users table
income	DECIMAL	64, 12	-
updated_time	DATETIME	-	-

Table 3.7 shows the details of provider income. This table store the information of service provider income by selling network services. Service provider can withdraw money to this income account from selling and transfer income to balance for buying other services.

Table 3.8 shows the ChoiceNet invoice table. Part of this table is duplicated from ipn table, however it is still necessary because it has three functions: (1) help users

Table 3.8: Invoice table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
date_created	DATETIME	-	-
service	FOREIGN-KEY	-	To service service table
buyer	FOREIGN-KEY	-	To accounts users table
amount	INT	1000	-
is_paid	TINYINT	1	-
is_active	TINYINT	1	-
number	VARCHAR	255	Unique

to handle their orders, (2) help services to manage their sales status, (3) after each payment, it can notify network setup part to provide services to users.

Table 3.9: Session table.

Name	**Type**	**Length**	**Description**
id	INT	11	Auto added, auto increment
session	INT	11	Random number as session key
start_time	DATETIME	-	Start time of session
end_time	DATETIME	-	End time of session
user	FOREIGN-KEY	-	To accounts users table
is_login	TINYINT	1	-
key	VARCHAR	128	The key of this session

Table 3.9 shows the ChoiceNet session table. This table is for the interaction with client application. It stores the session information and encryption key. Section 5 introduces more detail about the interaction between server and client.

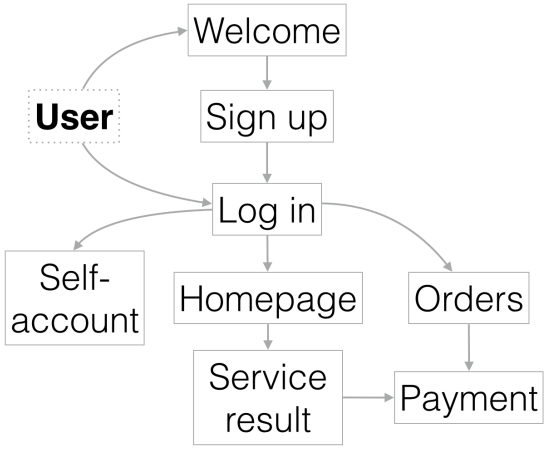
3.3 Web Interaction and Interface

Table 3.10: Account's accessibility to different page.

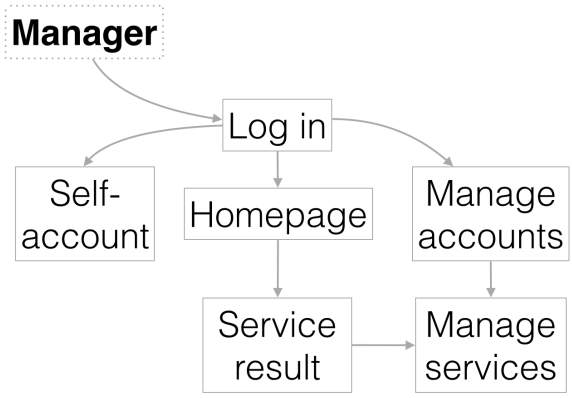
Web page	**User**	**Provider**	**Manager**	**Description**
Welcome	Y	Y	N	Instruction of using this web site
Sign up	Y	Y	N	User and provider submit information to obtain access
Log in	Y	Y	Y	To verify account identity
Self-account	Y	Y	Y	Account user can modify some attributes of their account
Homepage	Y	Y	Y	
Service results	Y	Y	Y	List results of services based on user search
Orders	Y	N	N	List all orders of user purchased
Payment	Y	N	N	User pay provider for service
Manage services	N	Y	Y	List all services provider uploaded
Upload service	N	Y	N	For provider uploading new service
Update service	N	Y	N	For provider modifying existing service
Manage accounts	N	N	Y	Web site manager manage all user and provider accounts

After proposed back-end database design, this section shows the operation of front-end. As stated in previous section, there are three types of accounts, each type of accounts has different accessibility to each web page. Account's accessibility is listed in table 3.10. *User* has lowest priority of the web site, but they can access all of the management pages, including manage services and manage accounts, except for manage self account page. *Provider* can manage and update their uploaded services

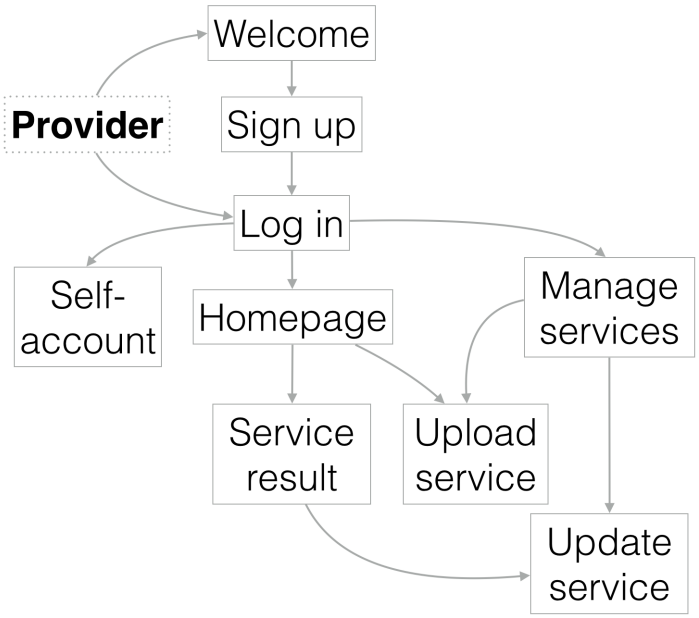
and create new services. *Manager* can manage all accounts and services with highest priority, but it is unnecessary for them to participate in economic activities.



(a) User operation flow.



(b) Web manager operation flow.



(c) Provider operation flow.

Figure 3.6: Operation flow for different accounts.

Figure 3.6a, 3.6c and 3.6b illustrate the typical operation flows of user, provider and manger respectively. From those operation flows, log in page is the central of

further operations. To utilize the full function of this web site, every entity has to sign up an account and log in. Guest user can only browse network services without any further steps, like choosing and paying for services or uploading and selling services. User and provider should have more interaction to keep each other information updated, such as user can leave feedback to provider and their services, provider can advertise their new and innovative services to users and both user and provider can watch the service status. All of those functions is finished by choiceNet sub-application.

3.4 Financial Transaction Integration

PayPal has a pretty decent API to support developer to integrate PayPal products. There are a lot of products for different business. Even though PayPal Payment Advanced and Pro have more functions such as customers paying without ever leaving the website and designing and hosting checkout pages for full control, PayPal Payment Standard can satisfy all of the financial demands of this web application. After comparisons of those products and to the functions of our web application, PayPal Payment Standard is the best choice among those products.

Every time a successful payment is made, PayPal is sending a *POST* request to this web site. After receive the IPN, this table create a new item contain information of the transaction. PayPal calls this process Instant Payment Notification (IPN) but it is known as webhooks. This method has some disadvantages because it drops users off to PayPal's website. However it's easy to implement and doesn't require SSL. Figure 3.7 show the authentication flow of IPN.

Paypal sub-application has one table, called *ipn*, which is a standard table for PayPal IPN. This table contains a large amount information of transaction including address, buyer and seller information and it is hard to handle. Therefore, we design our simple invoice table in choiceNet sub-application as a supplement.

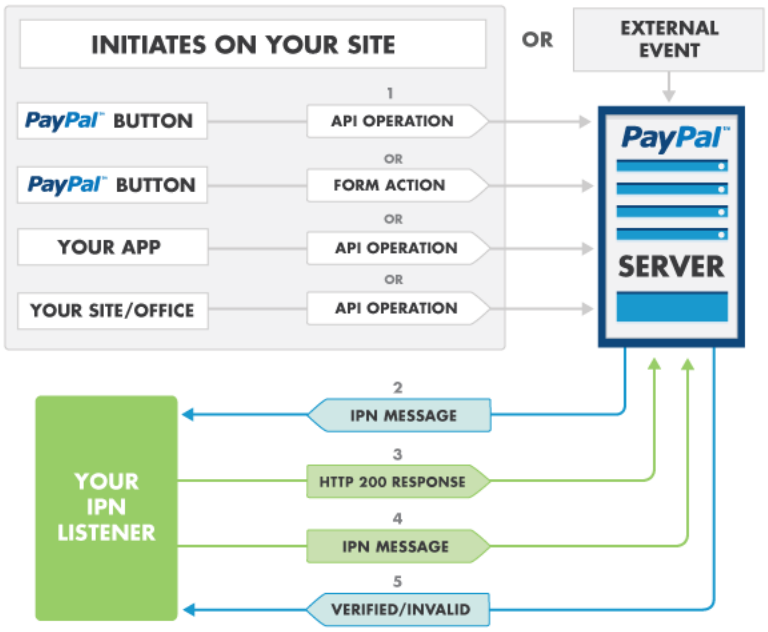


Figure 3.7: IPN auth flow [11].

3.5 Searching and Filtering Mechanism

It is important to have a proper design to handle user searching and filtering, because of large amount of services in this web application and excellent user experience demands. Traditional way is to query the server every time when user changes keywords or requirements of searching or filtering. Server have heavy burden to search, sort and filter results. Frequent query the server can also jam the server network traffic. Our method is to reduce the unnecessary query times to server and spare some searching and filtering work to client side, i.e. browsers.

Modern browser is a very powerful tool and can handle more complicated work than ever. Therefore, we propose that server handles searching and browser handles filtering results. To be specific, back-end server provides search result to client/browser and front-end browser filters and sorts the loaded searching results based on user requirements. Client will not query the server unless users changes their searching keywords. Data transmission between server and browser through Internet is also

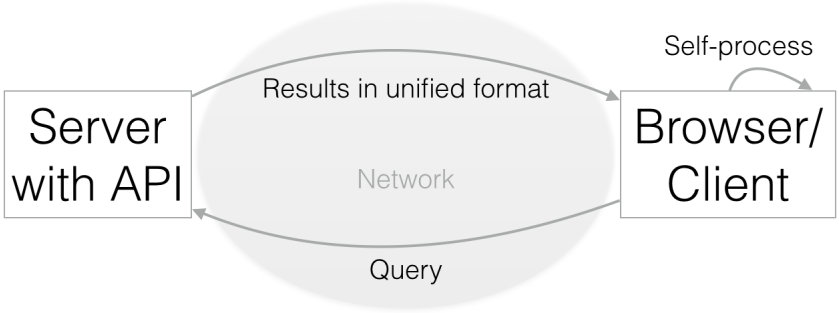


Figure 3.8: Server and browser interaction.

decreased. Users can experience more responsive and smooth operation in this web application. Browser in this web application plays an important role. It process part of the job which originally belongs to server and it only query the server when it is necessary. The relationship between server and browser is illustrated in figure 3.8.

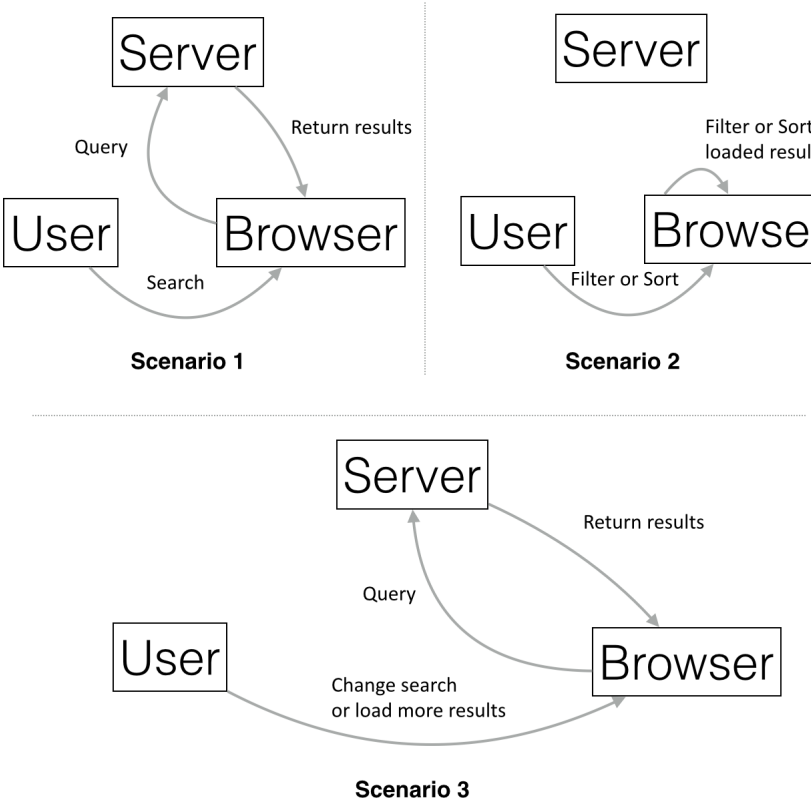


Figure 3.9: Searching and filtering scenarios.

Based on figure 3.8, we depict three scenarios of how user searching and filtering network services. Figure 3.9 shows those three scenarios. In scenario 2, server is free from browser query and browser processes data itself.

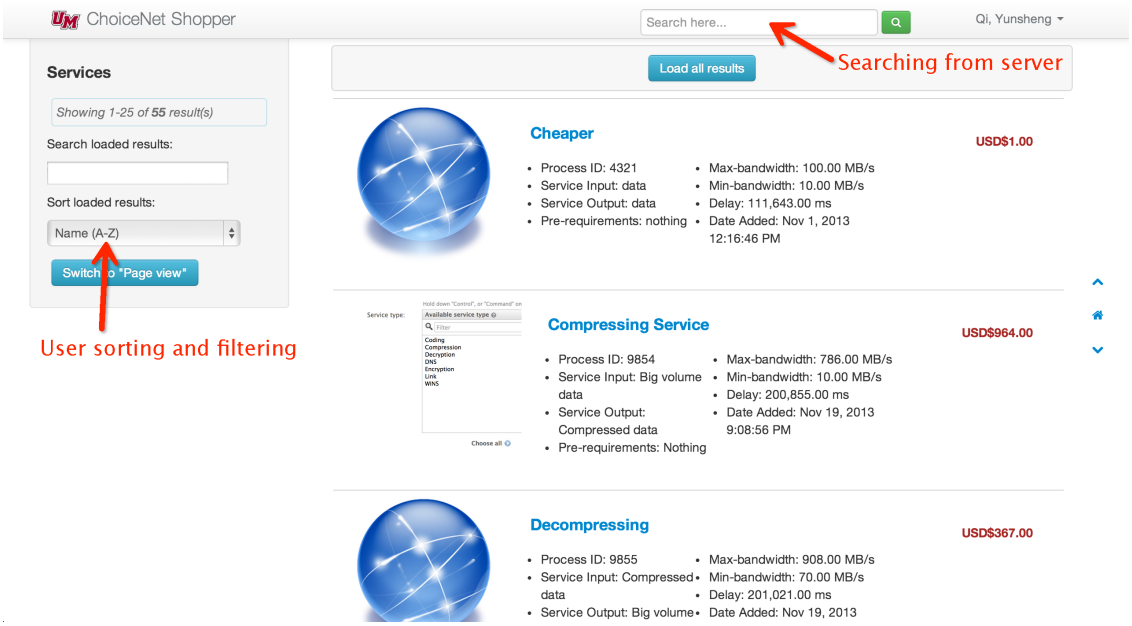


Figure 3.10: Searching and filtering interface.

Figure 3.10 shows searching interface. Every action in left grey column belongs to scenario 2 which take a large portion of user actions. Because grey column takes obvious position of the page, it can guide user full use of the "local utility" - browser. This interface use asynchronous web technique: AngularJS. Via this technique, web applications can communicate with a server asynchronously (in the background) without interfering with the display and behavior of the existing page. When users do some operations, like go to next page, without changing searching keyword, only network services data send to browser. The server do not need to send the large accessory files, like JavaScript and CSS, unless users start a new search.

3.6 Server-Client Interaction

At the client side, we have a client application which can run on user machine to detect any new network connection and provide choices to user based the connection. User than can select choice and pay for it without opening web browser or logging in PayPal. After transaction in web application, user will be served as they selected service. Figure 3.11 show the design of the interaction between server and client.

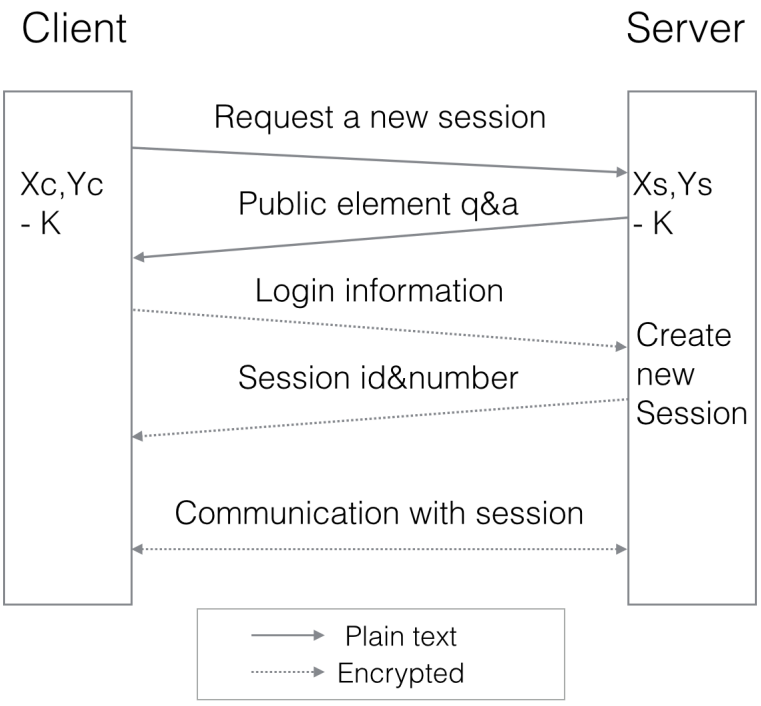


Figure 3.11: Interaction between server and client.

The initial interaction is executed by following steps. These steps includes "Secure Hash Function", "Symmetric Encryption" and "Diffie-Hellman Key Exchange Algorithm" in Chapter 2.

1. Server and client exchange cryptography data a and q
2. Client uses a and q to calculate client private key (X_c) and public key (Y_c)
3. Client sends request with Y_c to server

4. Server uses a and q to calculate server private key (X_s) and public key (Y_s)
5. Server calculates encryption key (K) by using X_s and Y_c
6. Server sends Y_s and session id to client
7. Client calculate encryption key (K) by using X_c and Y_s
8. Client sends encrypted user login information (username and hashed password) and session id to server to request login
9. Server deciphers and verifies user login information
10. Server sends the status of login to client
11. Server and client then can communicate with encryption

After initial interactions, user log in and session is established successfully. User can request choice of network from server. User than will get served after successfully payment by user account balance. User also can request refund if service did not set up or service is not satisfied.

3.7 Technique List

This section summarizes some techniques which are used in this web application.

- **MVC Design Pattern**: Model View and Controller is a software design pattern for implementing user interfaces by dividing software application into three interconnection parts.
- **Django 1.6**: It is the web framework implementing MVC design pattern.
- **Python 2.7.2**: It is the back-end programming language.
- **South 0.8**: It is a database migration tool to create new table or modify existing table.

- **Django-PayPal**: Django PayPal is a pluggable application that implements with PayPal Payments Standard to handle financial transaction.
- **AngularJS**: It is a front-end HTML enhancement to implement asynchronous technique.
- **Django REST Framework**: It is a toolkit to build Web APIs for asynchronous web pages.
- **Bootstrap**: It is a collection of compiled CSS, JS, and fonts.
- **MySQL**: Database.

CHAPTER 4

EVALUATION AND DISCUSSION OF RESULTS

In this chapter, we shows the results of the web application. First, we shows the implementation of database by MySQL. Second, pages of three types of accounts, user, provider and manager, are introduced. Comment system, PayPal integration and searching and filtering are shown in next three sections. Finally, section 6 shows the interaction between server and client.

4.1 Database

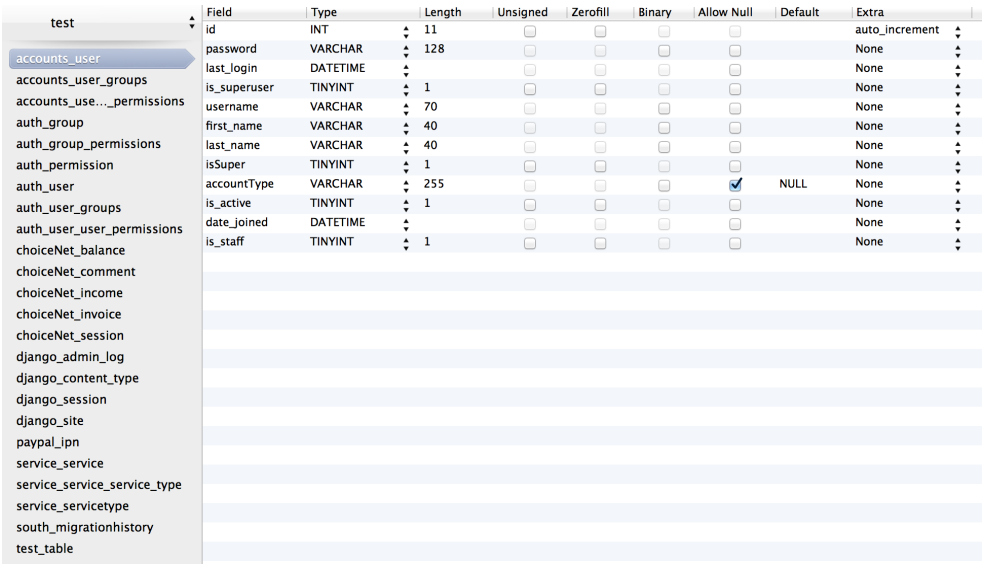


Figure 4.1: Database.

We use MySQL as our database. We create all of the tables which are discussed in last chapter. Figure 4.1 shows all tables for each sub-application and Django

standard tables. The name of each table begins with the name of corresponding sub-application.

Figure 4.2 shows the password status stored in database. The password is encrypted by SHA-256, so if the database is compromised, attacker still cannot decode it with limited resources.

id	password	last_login	is_superuser	username	first_name	last_name	isSuper	account
2	pbkdf2_sha256$12000$EdSA7xN...	2014-05-12 20:40:11	1	admin@admin.com	admin	admin	1	super
4	pbkdf2_sha256$10000$hIl3QpINv...	2013-09-23 20:07:49	0	user@user.com	user	user	0	user
22	pbkdf2_sha256$10000$WqMywS...	2013-09-21 02:40:02	0	test@test.com	test	test	0	user
24	pbkdf2_sha256$12000$ZhD2Xw3...	2014-05-12 22:30:47	0	yunsheng@umass.edu	Yunsheng	Qi	0	user
25	pbkdf2_sha256$10000$p5dp2hsE...	2013-09-21 03:00:34	0	test2@test.com	test2	test	0	user
26	pbkdf2_sha256$10000$xXTRe8a...	2013-09-21 03:05:15	0	test3@test.com	test3	test	0	user
27	pbkdf2_sha256$10000$lpsZuCP7...	2013-09-21 03:06:12	0	test4@test.com	test4	test	0	user
29	pbkdf2_sha256$10000$xc6oMrO...	2013-11-23 18:25:20	0	test50@test.com	test50	test	0	user

Figure 4.2: Password Storage.

4.2 Account Pages

4.2.1 Customer User

Figure 4.3 illustrates how the interface guides user to register and utilize the web. User have to first sign up with valid username (unique email address) and password. User first and last name are not required. After successfully signing up, user can log in this web and manage their account by changing user information and password.

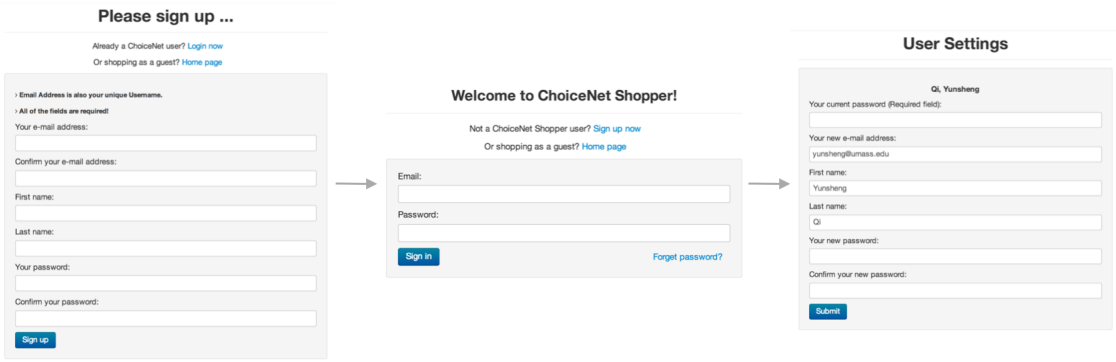
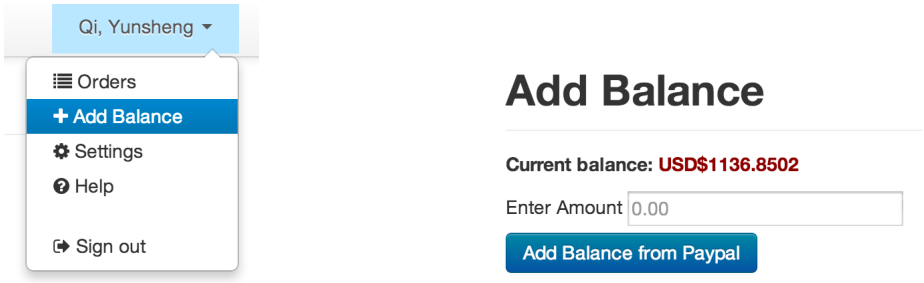


Figure 4.3: User registration process.

User entries are list on the top right corner of every page. Figure 4.4a shows the list of user entries. User can check past orders and add balance to their account. User can also go to account settings page and help page. Figure 4.4b is the page for user to add balance through PayPal. The current balance is also shown on this page.



(a) User entry list. (b) User add balance page.

Figure 4.4: User entry list and balance page.

Figure 4.5 is the interface for user to manage orders. In this page, user can comment and rate services, pay the unpaid orders and delete orders.

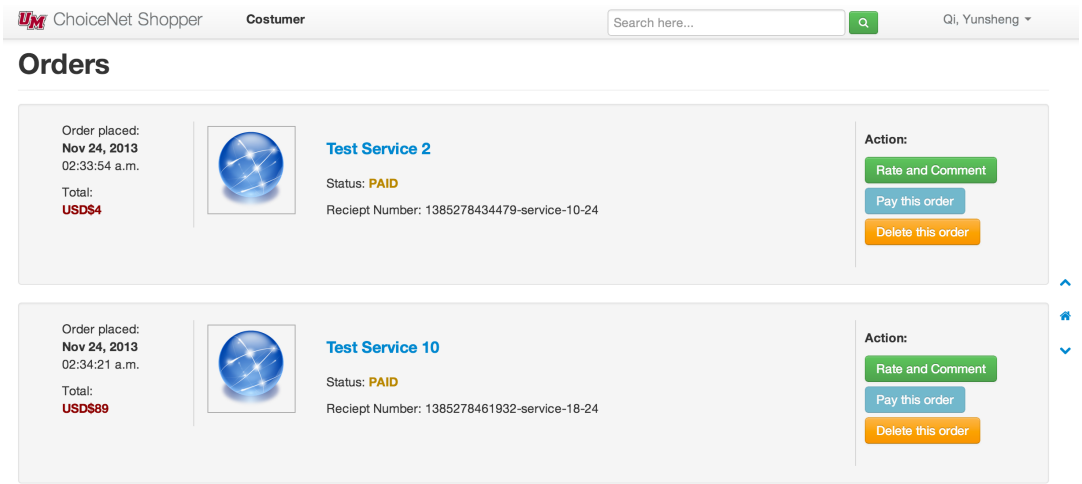
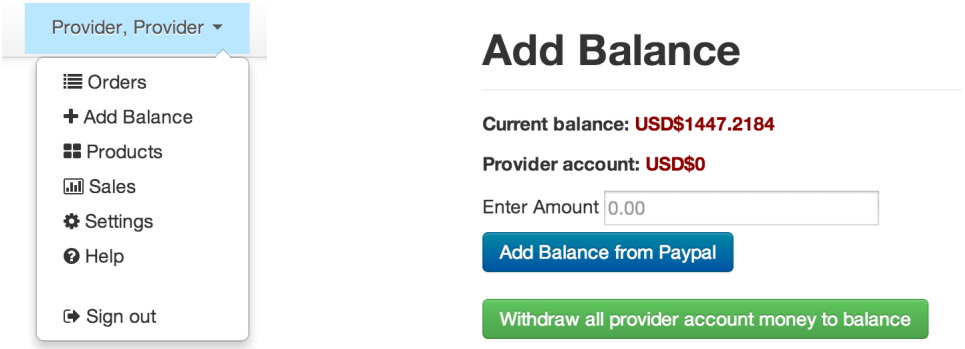


Figure 4.5: User orders management page.

4.2.2 Service Provider

Provider registration process and settings page are very similar to user's except for notifying web manager to add identity as a service provider. Provider entries are list on the top right corner of every page. Figure 4.6a shows the list of provider entries. Provider can check past orders, add balance to their account and go to account settings page and help page. Besides those functions same as user account, provider can also manage products (network services) which are uploaded by this provider. By checking sales page, provider can see the sales status of their services. Figure 4.6b is the page for provider to add balance. The current balance is also shown on this page. Provider can also add balance through PayPal or from their provider income account which are earned by selling services.



(a) Provider entry list.

(b) User add balance page.

Figure 4.6: Provider entry list and balance page.

Figure 4.7 shows the products list (services list) where provider can manage service by commenting and editing. In this page, service provider can also add new service.

Figure 4.8 is the page for provider to monitor their sales status. It also lists the total income.

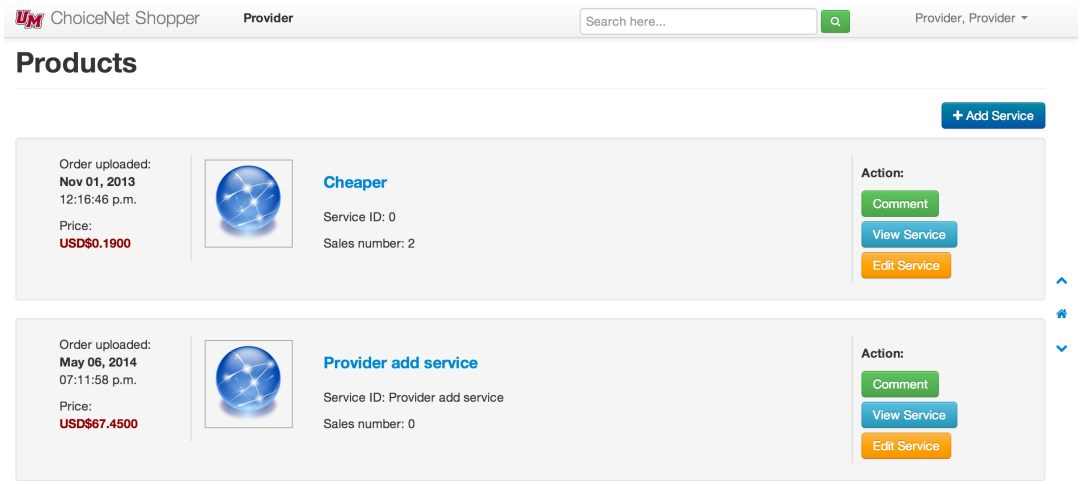


Figure 4.7: Provider service management page.

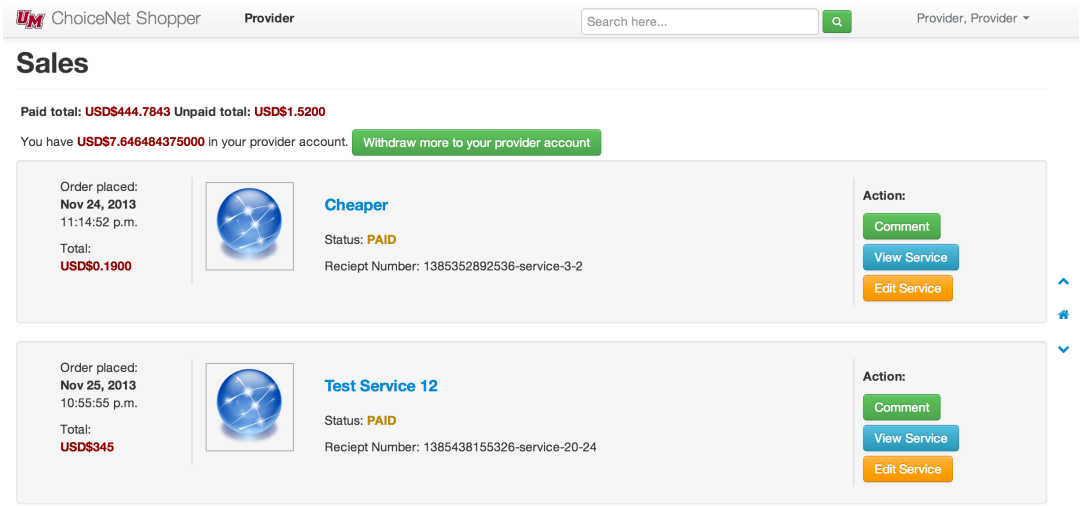


Figure 4.8: Provider sales management page.

4.2.3 Web Application Manager

Django Administration site allow super user to manage database remotely. This admin site is pretty decent to be used as web application manager management page. Figure 4.9 shows the interface of admin site. Manager can add or edit any account or service.

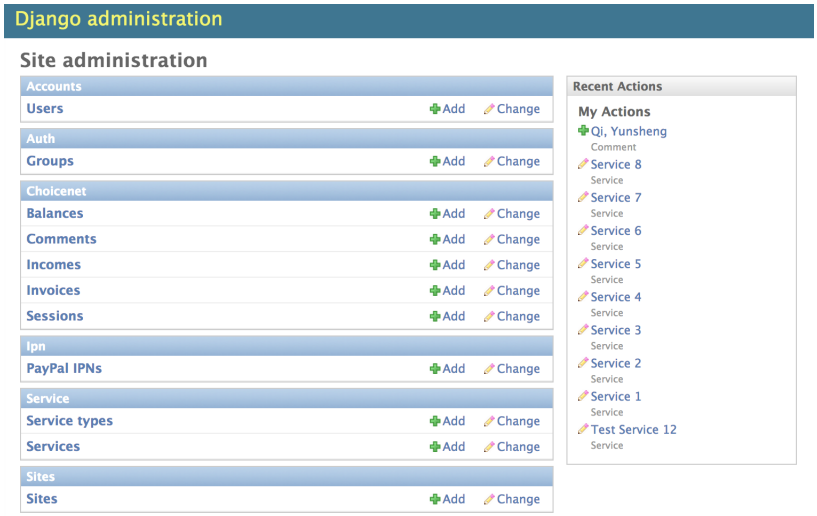
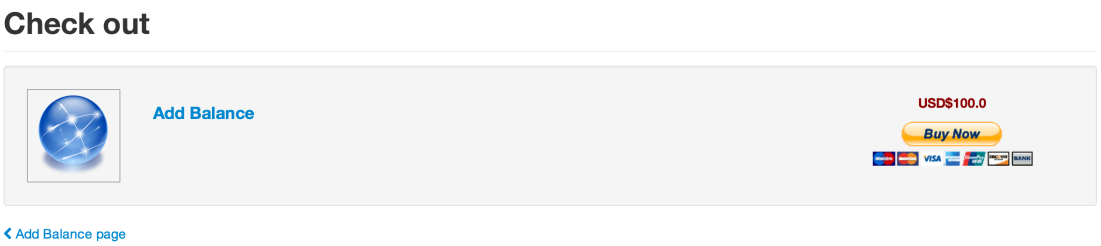


Figure 4.9: Web application manager admin site.

4.3 PayPal Integration

Because we are using PayPal Payment Standard, we cannot integrate payment page within this web application. Every time web application processes payment of adding balance, user has to redirect to PayPal web site with wrapped payment information. The wrapped PayPal payment information is including:

- **business**: The registered business entity email which receives the payment.
- **amount**: The number of items/services which are brought by user.
- **item_name**: Name of items/services.
- **invoice**: The unique invoice ID of this transaction.
- **notify_url**: The url for PayPal to notify the web site of successfully payment.
- **return_url**: The url for user to return the web site after transaction.
- **cancel_return**: The url for user to return the web site after user canceling transaction.



(a) User payment page.

PayPal IPN	Flag	Flag info	Invoice	Custom	Payment status	Created at
<IPN: Transaction 03W25844HL450103M>			1385418775155-service-53-2		Completed	Nov. 25, 2013, 5:24 p.m.

(b) PayPal IPN in database.

Date created 2	Buyer	Service	Number 3	Amount	Is paid	Is active
Nov. 25, 2013, 5:32 p.m.	admin, admin	Long name service long long long long long long long long	1385418775155-service-53-2	1		
Nov. 24, 2013, 11:14 p.m.	admin, admin	Cheaper	1385352892536-service-3-2	1		
Nov. 24, 2013, 2:33 a.m.	Qi, Yunsheng	Test Service 2	1385278434479-service-10-24	1		

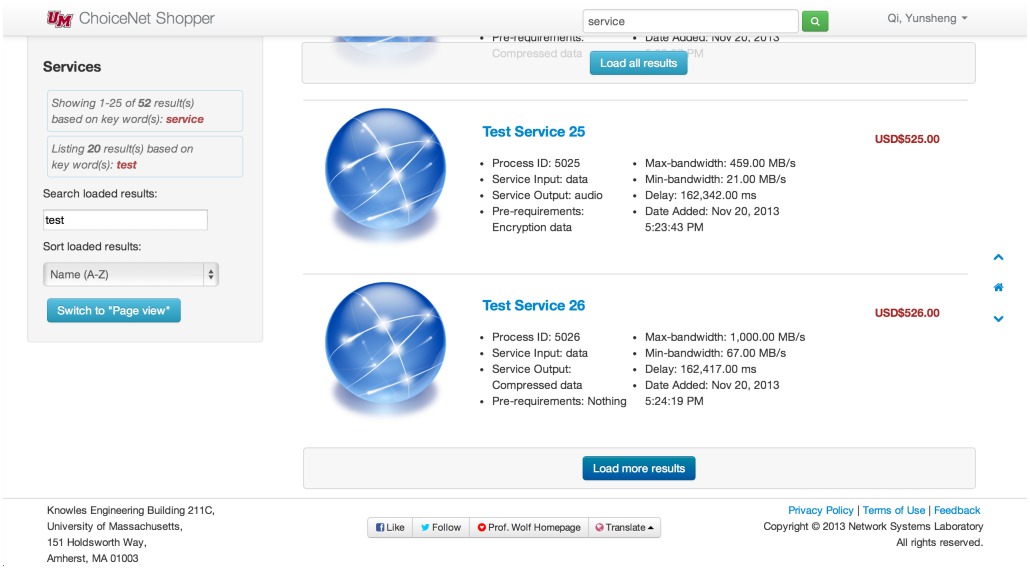
(c) Invoice in database.

Figure 4.10: PayPal integration.

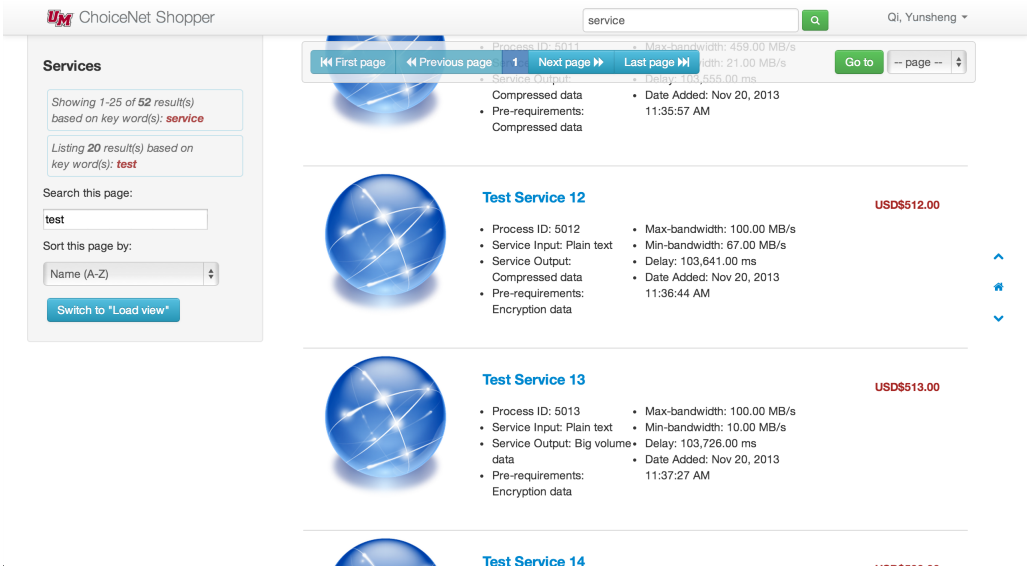
Figure 4.10a shows the payment page integrated with PayPal and figure 4.10b and figure 4.10c shows PayPal IPN and invoice in database after user successful payment. User will redirect to PayPal website to finish the payment by logging in their PayPal account and confirm payment. After payment on PayPal website, user will redirect back to this web application. At the same time, the web application get PayPal IPN and update the financial transaction information.

4.4 Searching and Filtering

We design two ways to list the searching results for user: *load view* (figure 4.11a) and *page view* (figure 4.11b). Load view let user load more results every time user scrolls to the bottom of the page or load all qualified results at once. Page view allows user to switch to different pages. Users can also filter or sort the existing results by interacting with left grey column.



(a) Load view.

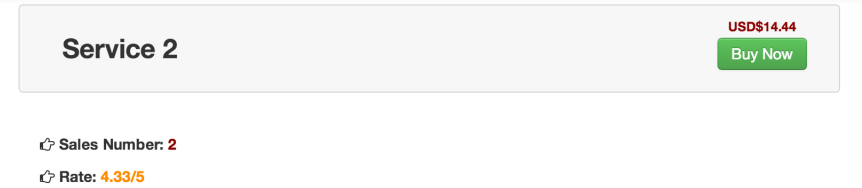


(b) Page view.

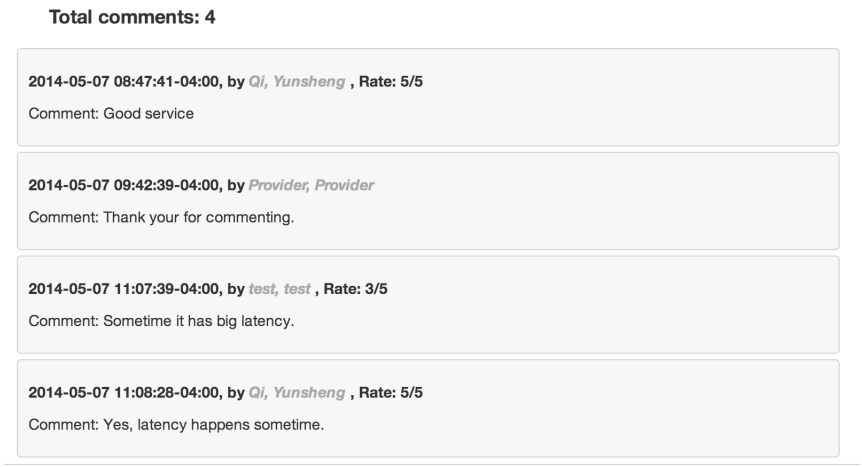
Figure 4.11: Two ways of search results visualization.

4.5 Rating and Comment System

After purchasing a network service, user can rate and comment the service on the orders page. Service provider also can comment the service they uploaded, but



(a) Rate and sales number.



(b) Comments.

Figure 4.12: Rating and comment system.

provider cannot rate. The average rate and all comments are shown on service detail page. Figure 4.12 shows rating and comment on service detail page.

4.6 Server-Client Interaction

Due to the requirements of Interaction between server and client, six interactions are implemented: request new session, user login, request network service, pay unpaid order, check payment status and request refund.

4.6.1 Request a new session

Figure 4.13 shows the public and encryption keys of server and client. The public keys of server and client are randomly generated and longer enough to protect data. The encryption key is a 64-bit key generated by logarithm of public and private key.

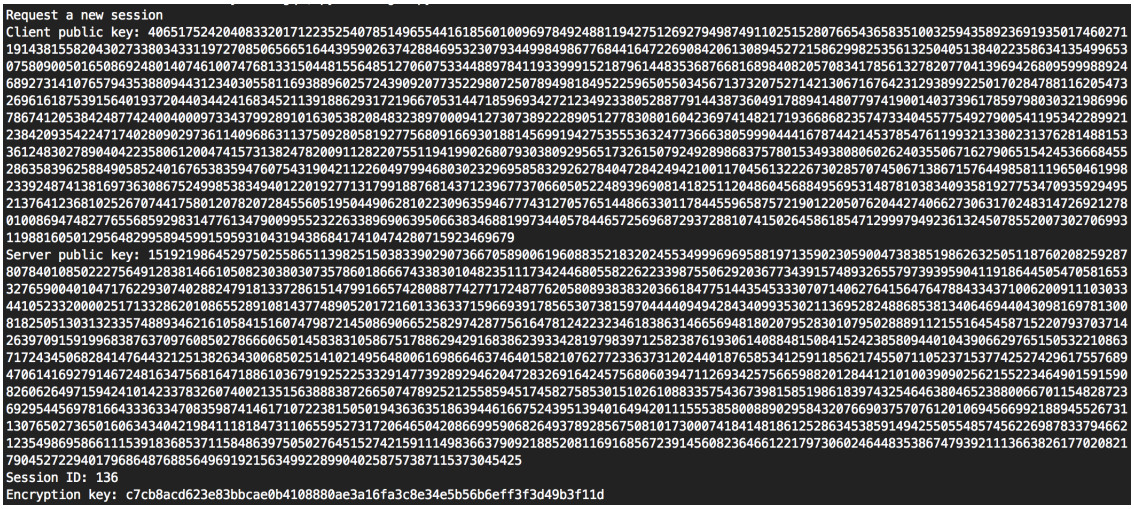


Figure 4.13: Request a new session.

The interaction variables and explanation contains two parts. URL is `...key/exchange/`. Data send to "key exchange":

$$send\_data = \{'publicKey' : publicKey\}$$

where $publicKey$ is the client public key for server to obtain cryptography key. Data received from "key exchange":

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where $is\_session$ is a boolean shows the session is set up or not, $expire$ is a boolean shows the session is expired or not, $data$ is plain text, if session is not created, $data = None$, and

$$data = \{'session\_id' : session\_id, 'publicKey' : publicKey\}$$

where $session\_id$ is the id of the session and $publicKey$ is the server public key to obtain cryptography key.

4.6.2 User Login

Figure 4.14 shows the interaction of user login. The password is hashed and protected to send to server.


```
User login
Password (plain text): yunsheng
Password (hashed text): dd22e6d874291ec9a1320da2b49e6665bdf180e3
Client plain text data: {'username': 'yunsheng@umass.edu', 'password': 'dd22e6d874291ec9a1320da2b49e6665bdf180e3'}
Client cipher data: 0b10100110000000011100100110110101100101011001100011011111011100001101100101101001011110110001110001000110101001111101011111000101
00001001110100010000111100110111111111000101001000001110101001100010110101011100101101101010010000011111011111100101000001100011100001001010001101000100
10010101010011010110111101010000000100010101001000101001011010001001000000000100010001000000010111101001101011011010010111000001000011110010000110010
01110000010101100001000101101001011011111010001100100100100010110110011100111110001010100010000000100101001001110101100100101011101111101101010011011 00
0011000000000000011110011000010000000010110011010011110010010000011000010101001000101110100000100010110011011011010100100100110010001101011
Server cipher data: {u'expire': False, u'data': u'0b10100110000000010111000001110000111000010101111001000101111011100011101011110000000001011101101110
0011100110101100111011000010110110101010011001001001010101111110101001101011001011011011000000001010101111101110000111000101011100110110111000011000011
110010100100001110011000100101111110110101010110010110000000001000100111011100101110000111000101101111001111000001010010101001001010100110101100111 0100
0000011101010110010110000000010010010011001110111100010100000010110110001001101001100110000101111100000100110001100011011010110000000101000011001011101
101101101', u'is_session': True}
Deciphered server data: {u'session': 94614787, u'balance': u'1136.850195312500', u'success': True}
```


Figure 4.14: User login.

URL is `...request/new/session/`. Data send to "user login":

$$send\_data = \{'data' : data, 'session\_id' : session\_id\}$$

where *session_id* is the id of session and *data* is cipher text, and

$$data = \{'username' : username, 'password' : password\}$$

where *username* is the email address of user and *password* is the hashed password. Data received from "user login":

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where *is_session* is a boolean shows the session is set up or not, *expire* is a boolean shows the session is expired or not and *data* is cipher text, if session is not created, $data = None$, and

$$data = \{'success' : success, 'session' : session\}$$

where *success* is a boolean shows login success or not and *session* is the number of session.

4.6.3 Request Network Service

Figure 4.15 is for client to request a network service from server. Client must send service ID to server and other verification information. Server returns a url to redirect user to make a payment by PayPal.

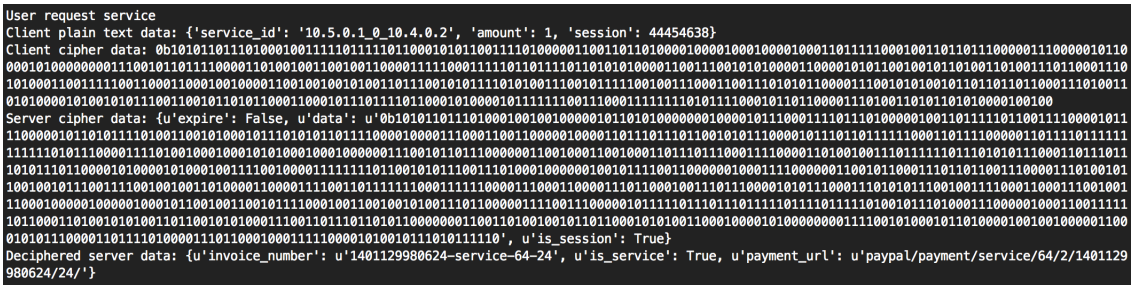


Figure 4.15: Request network service.

URL is `...request/service/`. Data send to "request service":

$$send\_data = \{'data' : data, 'session\_id' : session\_id\}$$

where *session_id* is the id of session and *data* is cipher text, and

$$data = \{'service\_id' : service\_id, 'session' : session, 'amount' : amount\}$$

where $service_id$ is the id of service request, *session* is the number of session and *amount* is the amount of request service. Data received from "request service":

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where *is_session* is a boolean shows the session is set up or not, *expire* is a boolean shows the session is expired or not and *data* is cipher text, if session is not created, $data = None$, and

$$data = \{'is\_service' : is\_service, 'invoice\_number' : invoice\_number,$$
$$'payment\_url' : payment\_url\}$$

where *is_service* if service does not exist, return false, *invoice_number* is the invoice of transaction and *payment_url* the url redirect user to make payment by PayPal (the full url must be $url\_root + payment\_url$).

4.6.4 Pay Unpaid Order

User can pay unpaid order directly from client app. Figure 4.16 shows this interaction.

Figure 4.16: Pay unpaid order.

URL is `...pay/order/`. Data send to "pay unpaid order":

$$send\_data = \{'data' : data, 'session\_id' : session\_id\}$$

where *session_id* is the id of session and *data* is cipher text, and

$$data = \{'invoice\_number' : invoice\_number, 'session' : session\}$$

where *invoice_number* is the order to pay and *session* is the number of session. Data received from "pay unpaid order":

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where *is_session* is a boolean shows the session is set up or not, *expire* is a boolean shows the session is expired or not and *data* is cipher text, if session is not created, $data = None$, and

$$data = \{'is\_invoice' : is\_invoice, 'invoice\_number' : invoice\_number,$$
$$'payment\_url' : payment\_url, 'previous\_paid' : previous\_paid\}$$

where *is_invoice* if invoice does not exist, return false, *invoice_number* is the invoice of transaction, *payment_url* the url redirect user to make payment by PayPal (the full url must be $url\_root + payment\_url$) and *previous_paid* if the order is already paid, return true.

4.6.5 Check Payment Status

User can check payment status after finished a payment by PayPal from client app. Figure 4.17 shows this interaction.

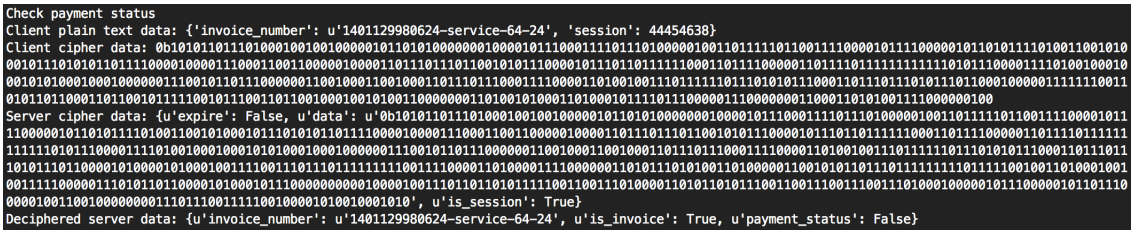


Figure 4.17: Check Payment Status.

URL is `...check/payment/status/`. Data send to "request refund":

$$send\_data = {'data' : data, 'session\_id' : session\_id}$$

where *session_id* is the id of session and *data* is cipher text, and

$$data = \{'invoice\_number' : invoice\_number, 'session' : session\}$$

where *invoice_number* is the order to pay and *session* is the number of session. Data received from "request refund":

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where *is_session* is a boolean shows the session is set up or not, *expire* is a boolean shows the session is expired or not and *data* is cipher text, if session is not created, $data = None$, and

$$data = \{'is\_invoice' : is\_invoice, 'invoice\_number' : invoice\_number,$$
$$'payment\_status' : payment\_status\}$$

where *is_invoice* if invoice does not exist, return false *invoice_number* is the invoice of transaction and *payment_status*: the status of payment.

4.6.6 Request Refund

User can request refund directly from client app if user does not get served or the service is not satisfied. Figure 4.18 shows this interaction.

```
Request refund
Client plain text data: {'invoice_number': u'1401129980624-service-64-24', 'session': 44454638}
Client cipher data: 0b101011011101000100100100000101101010000000010000101110001111011101000001001101111101100111100001011110000010110101111010011001010
0010111010101101111000010000111000110011000000100001101110111011001010111000010111011011111100011011110000011011110111111111111010111000011110100100010
00101010001000100000011100101101110000001100100011001000110111011100011110000110100100111011111110111010101110001101110111010111011000100000111111 0011
010110110001101100101111100101110011011001000100101001100000001101001010001101000101111011100000111000000001100011010100111100000000100
Server cipher data: {u'expire': False, u'data': u'0b101011011101000100100100000001011101100101100000111100111110101000111011111001110001110011101011101
11001011011100000110010011010000000111001111101111111001000111010011111010001000001111101011000001110111110011010000000111000010100010101001001100101111
0001110110001111111111110010010101001100100000101000101001000101110000011', u'is_session': True}
Deciphered server data: {u'is_invoice': True, u'is_refund': False}
```

Figure 4.18: Request refund.

URL is `...request/refund/`. Data send to “request refund”:

$$send\_data = 'data' : data, 'session\_id' : session\_id$$

where $session\_id$ is the id of session and $data$ is cipher text, and

$$data = \{'invoice\_number' : invoice\_number, 'session' : session\}$$

where $invoice\_number$ is the order to pay and $session$ is the number of session. Data received from “request refund”:

$$received\_data = \{'is\_session' : is\_session, 'expire' : expire, 'data' : data\}$$

where $is\_session$ is a boolean shows the session is set up or not, $expire$ is a boolean shows the session is expired or not and $data$ is cipher text, if session is not created, $data = None$, and

$$data = \{'is\_invoice' : is\_invoice, 'is\_refund' : is\_refund\}$$

where $is\_invoice$ if invoice does not exist, return false and is_refund return false, if request is unsuccessful.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter first draws the conclusion of the web application. Although the web application meets the requirements ChoiceNet and every function of proposed design is implemented, there are still some improvements on this project which is discussed in the second section.

5.1 Conclusion

This thesis introduces the design and implementation of the network service marketplace. At first, the slow commercial deployment of multicast motivates us to this research, called ChoiceNet. ChoiceNet not only can solve the problem of multicast, but also is the most suitable option for future Internet. As part of ChoiceNet, we decide to use a web application to implement the marketplace. Based on the requirements of ChoiceNet, a list of problems are stated.

As the foundation of this thesis, we introduce related research work on ChoiceNet. The definition of choice and the whole picture of ChoiceNet are first introduced. The definition and composition of service are the basis of designing database and decide how the service stored in database. An economy plane is added on the top of control plane which is existing in current network. The function of economy plane is also shown. Security is another concern of communication between server and client, so useful theories and knowledge are depicted.

Based on related work, we propose the design of the web application. We first go through the overview of the design and then introduce the details by following orders:

database, interface and interaction, financial transaction integration, searching and filtering mechanism and server-client interaction.

At last, the results are shown. We implement three types of accounts to act different roles in ChoiceNet, including customer user, service provider and web manager. We use PayPal integration to implement the principle-"Vote with Your Wallet", use searching and filtering to implement "Encourage Alternative" principle and use rating and comment system to implement "Know What Happened". The interaction between server and client is shown with security protection.

5.2 Future Work

Though this thesis finished all proposed work, there are still some work can be done in future to improve the whole research. This web application is designed for a research, so it is far from commercial deployment. User interface (UI) of all three types of accounts should be more distinguished from each other and each UI should be designed friendly for each type of account. The database should be expanded to handle more complicated network service. It is better to have more payment methods other than only PayPal. The connections of client-server and web-planner are still weak and more work need to be done to strengthen the connection.

BIBLIOGRAPHY

[1] Anderson, Tom, Birman, Ken, Broberg, Robert, Caesar, Matthew, Comer, Douglas, Cotton, Chase, Freedman, Michael J, Haeberlen, Andreas, Ives, Zachary G, Krishnamurthy, Arvind, et al. The nebula future internet architecture. In *The Future Internet*. Springer, 2013, pp. 16–26.

[2] Burbeck, Steve. Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). *Smalltalk-80 v2 5* (1992).

[3] Chen, Xinming, Dwaraki, Abhishek, Cai, Hao, and Wolf, Tilman. Specification and composition of network services in future internet architectures. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop* (2012), ACM, pp. 45–46.

[4] Clark, David D, Wroclawski, John, Sollins, Karen R, and Braden, Robert. Tussle in cyberspace: defining tomorrow's internet. In *ACM SIGCOMM Computer Communication Review* (2002), vol. 32, ACM, pp. 347–356.

[5] Diot, Christophe, Levine, Brian Neil, Lyles, Bryan, Kassem, Hassan, and Balensiefen, Doug. Deployment issues for the ip multicast service and architecture. *Network, IEEE 14*, 1 (2000), 78–88.

[6] Dwaraki, Abhishek, and Wolf, Tilman. Service instantiation in an internet with choices. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on* (2013), IEEE, pp. 1–7.

[7] HOLOVATY, A, and KAPLAN-MOSS, J. The django book: Version 2.0. *The Django Book 16* (2009).

[8] Miniwatts Marketing Group, Internet World Stats. World internet users and population stats. `http://www.internetworldstats.com/stats.htm`, June 2012.

[9] Nagurney, Anna, Li, Dong, Wolf, Tilman, and Saberi, Sara. A network economic game theory model of a service-oriented internet with choices and quality competition. *NETNOMICS: Economic Research and Electronic Networking* (2013), 1–25.

[10] Nagurney, Anna, and Wolf, Tilman. A cournot–nash–bertrand game theory model of a service-oriented internet with price and quality competition among network transport providers. *Computational Management Science* (2013), 1–28.

[11] PayPal. Instant payment notification: Getting started. `https://developer.paypal.com/docs/classic/ipn/gs_IPN/`, Jan. 2014.

[12] Rouskas, George N, Baldine, Ilia, Calvert, Kenneth L, Dutta, Rudra, Griffioen, Jim, Nagurney, Anna, and Wolf, Tilman. Choicenet: Network innovation through choice. In *ONDM* (2013), pp. 1–6.

[13] Shanbhag, Shashank, Huang, Xin, Proddatoori, Santosh, and Wolf, Tilman. Automated service composition in next-generation networks. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on* (2009), IEEE, pp. 245–250.

[14] Stallings, William. Network security essentials: Applications and standards (2000).

[15] Wikipedia. Ip multicast. `http://en.wikipedia.org/wiki/IP_multicast`, Jan. 2014.

[16] Wolf, Tilman, Griffioen, James, Calvert, Kenneth L, Dutta, Rudra, Rouskas, George N, Baldine, Ilia, and Nagurney, Anna. Choice as a principle in network architecture. *ACM SIGCOMM Computer Communication Review 42*, 4 (2012), 105–106.

[17] Wolf, Tilman, Griffioen, Jim, Calvert, Ken, Dutta, Rudra, Rouskas, George, Baldin, Ilya, and Nagurney, Anna. *ChoiceNet: Toward an Economy Plane for the Internet*. December, 2013.

[18] Zhang, Lixia, Estrin, Deborah, Burke, Jeffrey, Jacobson, Van, Thornton, James D, Smetters, Diana K, Zhang, Beichuan, Tsudik, Gene, Massey, Dan, Papadopoulos, Christos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC* (2010).