OPTIMAL SOFTWARE STRATEGIES IN THE PRESENCE OF NETWORK EXTERNALITIES

By

YIPENG LIU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

To my Mom, Peizhen Wu

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to all those who gave me the possibility to complete my education. I am deeply indebted to my advisors, Dr. Hsing K. Cheng and Dr. Gary J. Koehler. Their willingness to patiently share their expertise coupled with a gentle guiding hand and high expectations inspired me to seek excellence. My thanks also go to my committee members Dr. Steven M. Shugan, Dr. Juan (Jane) Feng and Dr. Subhajyoti Bandyopadhyay for freely sharing their time and for their constructive comments and suggestions on my dissertation. I am especially grateful to Dr. Haldun Aytug and Dr. Asoo J. Vakharia for their kind help in my life and for giving me opportunity to work here at the University of Florida and providing the resources necessary for me to achieve my goals.

I would like to extend my thanks to all my colleagues in the Ph.D. program for their unwavering moral support and friendship throughout my years in the program, especially Ms. Shawn F. Lee and Ms. Patricia Brawner. Thanks also go to my friends who have been patient, understanding and always encouraging during the many triumphs and failures along the way. I especially appreciate my dear friends Mr. Sean Marston and Ms. Hong Guo for their tireless support.

The episode of acknowledgement would not be complete without expressing my sincere thanks to my wife, Xiaoxiao Hong, with the deepest sense of gratitude. She rendered me enormous help throughout the whole time of my study and I will never forget every moment in which she helped me overcome spiritual and financial difficulties with her undying love.

Finally I am forever indebted to my parents, particularly my mom, Peizhen Wu, for always believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

OPTIMAL SOFTWARE STRATEGIES IN THE PRESENCE OF NETWORK EXTERNALITIES

By

Yipeng Liu

August 2009

Chair: Hsing K. Cheng
Cochair: Gary J. Koehler
Major: Business Administration

Network externalities or alternatively termed network effects are pervasive in computer software markets. While software vendors consider pricing strategies, they must also take into account the impact of network externalities on their sales. My main interest in this research is to describe a firm's strategies and behaviors in the presence of network externalities and I look into three software strategy problems specifically — the free trial strategy problem, pricing strategy problem and compatibility strategy problem.

In Chapter 2, I investigate the influence of network effects on the free trial strategies of a software firm under a monopoly scheme. I build an analytical model to examine the tradeoff between the effects of reduced uncertainty and demand cannibalization, and aim to uncover conditions under which software firms should introduce a time-locked free trial. I find that time-locked free trial outperforms the limited version free trial by bringing more profit to the software firm when consumers' prior belief is low and the network effect intensity of the software is modest.

In Chapter 3, I discuss a software firm's optimal pricing strategies under the coalescing effect of piracy and word-of-mouth. I extend an empirical software diffusion model and find the

optimal prices of a software product throughout its life cycle. In the settings of only one price change permitted, I show that a market penetration strategy dominates a skimming strategy when the demand of innovators is low, but loses ground and is replaced by the skimming pricing strategy as the demand of innovators increases.

In Chapter 4, I examine the impact of network externalities on the competition between open source software (OSS) and proprietary software. I find that when the market is fully covered, the installed base and the profit of proprietary software increase at the expense of decreasing user base for OSS in the presence of network externalities. This competitive imbalance becomes more pronounced when a third product, commercially supported OSS, joins the competition. Furthermore, I find that OSS venders always have the most incentive to make its product compatible with proprietary software.

CHAPTER 1
INTRODUCTION

Network externalities exist when the value of consuming a particular product increases in the number of consumers that use the same product. Many information goods such as software and online content demonstrate salient consumption externalities: the larger the user base, the greater the user perceived value. The presence of network externalities has proven to be problematic in traditional economic theory as manifested by the extensive research on this subject. However, little formal modeling of a firm's response to network effects and standardization in the presence of these effects has been completed to date. Therefore, the objectives of this research are many fold: (1) formally analyze the strategies of software firms to the evolvement and presence of network effects; (2) determine if there are commitment mechanisms that a firm can use to facilitate the introduction and diffusion of a software product and modify the downward pressure on price of such goods over time; and (3) investigate the compatibility strategies of open source and proprietary software venders and the impact of network externality on the competition between them.

In general, there are two types of network exteralities – direct and indirect. A direct network effect arises when there is "a direct physical effect of the number of purchasers on the value of the product" (Katz and Shapiro 1985, p. 424). Katz and Shapiro (1994) define networks with a direct physical effect if the value of joining this network depends on the umber of other consumers who join by adopting the same or campatible product. Not surprisingly the canonical examples are communication networks such as telephone, email and facsimile standards. Indirect externalities, on the other hand, often refer to the increasing utility derived from users of complementary goods. For example, users of Macintosh computers are better off the greater the number of consumers who purchase Macs because the larger the numer of Mac users the greater

the deamnd for compatible software, which if matched will lead to lower prices and/or a greater variety of software which makes all Mac users better off. In this study, I choose to focus on the direct network externality when considering open software strategies. This is because other than operating systems, one seldom observes this hardware/software complimentary paradigm applying to the existing software market. In addition, most software users make their decisions to join a software network based on their private benefits and do not take into account that others on the network are also made better off by their decision to join, which is a typical attribute for networks characterized by direct network effects (Church et al. 2003)..

This dissertation is organized as a collection of three essays, each of which covers one of several aspects of the entire study. Each chapter corresponds to an essay which is complete by itself. Due to this self-contained style of preparation, some redundancies across chapters may arise. This section is intended to give an outline of this dissertation.

In Chapter 2, I develop an analytical model to investigate optimal software free trial strategies. Free trial offers are used by many companies to sell various products ranging from books, CDs, and magazines to Internet access. Offering free trials can be an effective strategy to promote the diffusion of experience goods. There are two general forms of free trial software in the market – a free fully functional version with limited trial time (termed time-locked free trial) and a free "demo" version with limited functionalities (commonly referred to as limited version free trial). Companies in the software industry seem to favor either time-locked free trial or limited version free trial, but rarely the combination of both. Since a consumer using the time-locked free trials has access to full functionality of the software, the time-locked free trial is suitable for reducing consumers' uncertainty about functionality offered by the software. Offering time-locked free trials, however, can have a negative impact on the software firm.

When the firm introduces a time-locked free trial along with its commercial product, it risks losing part of the demand for its commercial product as consumers who have only short-term usage can now utilize the time-locked free trial without buying the commercial product. Further, if the trial users decide not to buy the product after the trial period, no positive network effect can be derived from those trial users. Unlike time-locked free trial, limited version free trial has the advantage of capturing the network effect from both trial users and the buyers. Partial demand of the commercial software, however, can still be cannibalized by the limited version free trial. For example, some consumers may find it sufficient to use only the functionalities provided in the trial instead of purchasing the commercial product. Should the software firm offer time-locked free trial to take advantage of the reduced consumers' uncertainty of its product or forgo the free trial to avoid the negative "cannibalization" effect? Thirty-day time-lock trial periods seem to be the most common length for a free trial, but is it truly an optimal length of free trial time that maximizes the software producer's profit? A major objective of Chapter 2 is to answer these critical questions.

In Chapter 3, I develop an analytical model that employs the empirical findings on software diffusion by Givon, et al (1995) to examine optimal pricing strategies for a software product under two coalescing effects of piracy and word-of-mouth through the software product's entire life cycle and evaluate how the change of underlying parameters (e.g., the demand of the innovators, the word-of-mouth effect, the legal conversion rate, etc.) will impact the software firm's pricing decision. Specifically, how to price software in the presence of piracy and word of mouth effect is a critical problem for software firms. A higher than optimal price results in more piracy and revenue loss as "software too expensive" is found as the major reason for consumer piracy (Cheng et al. 1997). Further, it leads to an undesired consequence of a

smaller installed base and slower diffusion of the software product. A lower than optimal price has the benefit of discouraging piracy at the sacrifice of profit loss. While the industry has been engaged in piracy proofing, academic literature has largely focused on pricing issues without adequately addressing such digital product characteristics as the network effect. The word-of-mouth influence on the diffusion of the software in essence is equivalent to the positive network externality effect in economics (Ellison and Fudenberg 1995, Cabral et al. 1999, Godenberg et al. 2003). In addition, the legitimate demand for digital experience goods, albeit with variable price boundaries, piracy deterrents and quality/feature preferences, has largely been ignored. The purpose of Chapter 3 is to fill this gap.

In Chapter 4, I turn my attention to the competition dynamics between OSS and proprietary software (PS). Unlike prior literature, I choose to focus on the impact of network externalities on the competition rather than the result of competition between open source and proprietary software and the strategic choices and incentives for compatibility between OSS and proprietary software venders – a topic majorly overlooked in previous studies on open source software. To survive and win the battle of market shares, OSS and proprietary software vendors may strategically choose to have their products be compatible or incompatible with its rival software. For example, Windows, a proprietary software product, is incompatible with Linux, an open source software product. In the case of web browsers however, Internet Explorer is completely compatible with its open source rival Mozilla Firefox in the sense that files created for IE users can be used by Firefox users and vice versa. Different compatibility strategies will lead to different network sizes, and thus result in different network generated values and profits for both OSS and proprietary software venders, which in turn present a series of research questions: Does the positive network effect always have a positive impact on both OSS and

proprietary software? If not, which party will benefit and which party will be hurt? How would the choice of compatibility affect network externality's impact? Furthermore, which party has the most incentive to make its product compatible with its rival in the presence of network externality? A major objective of Chapter 4 is to answer these critical questions.

The discussion of the results and future work is concluded in Chapter 5.

CHAPTER 2
SOFTWARE FREE TRIAL: TO TIME LOCK OR NOT

Preliminaries

Free trial offers are used by many companies to sell various products ranging from books, CDs, and magazines to Internet access. Offering free trials can be an effective strategy to promote the diffusion of experience goods. Unlike physical goods, some information goods, such as software, movies and music have a somewhat peculiar property where a consumer may not be able to assess the goods accurately and reliably before the consumption of the goods. Sometimes, the process of evaluating such information goods itself is the very process of consumption. This uncertainty of product functionality reduces a consumer's willingness to adopt the product and is considered by many a source of market failure. Due to the negligible marginal production cost nature of digital goods, offering free trial to help reduce consumers' uncertainty is more favorable to the digital goods vendors than physical goods makers as evidenced by the prevalence of free trials from software companies nowadays.

There are two general forms of free trial software in the market – a free fully functional version with limited trial time (termed time-locked free trial) and a free "demo" version with limited functionalities (commonly referred to as limited version free trial). Companies in the software industry seem to favor either time-locked free trial or limited version free trial, but rarely the combination of both. For instance, Ilium Software offers a 30-day free trial plan for its newly released NewsBreak, a media software product that delivers industry news, weather forecast as well as stocks information to pocket PC or Smart phones. Apple Inc. offers its recently released professional photo processing software Aperture with all features included (e.g., Advanced Color Controls, Onscreen Controls etc.) free of charge, but only for thirty days. To exploit the benefit from positive network effect, some software firms offer free trial version

software with limited functionalities for unlimited time (i.e., without a time lock) along with their commercial products to build up their user base. For example RealNetworks gives away its RealPlayer, which in essence is a “light” version of RealPlayer Plus that offers many more advanced features such as advanced CD burning, movie-on-demand and live music stations, in order to establish a sizable network of users for the benefits of RealPlayer Plus. Another example is PQ Computing’s PQ DVD, a video converter suite that converts DVD movies to iPod video. PQ DVD’s free evaluation version offers almost identical functionalities of the full version except that it only converts the first two minutes of the movie.

Since a consumer using the time-locked free trials has access to full functionality of the software, the time-locked free trial is suitable for reducing consumers’ uncertainty about functionality of the software. Offering time-locked free trials, however, can have negative impact on the software firm. When the firm introduces a time-locked free trial along with its commercial product, it risks losing part of the demand for its commercial product as consumers who have only short-term usage can now utilize the time-locked free trial without buying the commercial product. Further, if the trial users decide not to buy the product after the trial period, no positive network effect can be derived from those trial users. Unlike time-locked free trial, limited version free trial has the advantage of capturing the network effect from both trial users and the buyers. Partial demand of the commercial software, however, can still be cannibalized by the limited version free trial. For example, some consumers may find it sufficient to use only the functionalities provided in the trial instead of purchasing the commercial product. This paper will first focus on analyzing the time-locked free trial strategy and then address the issue of which software free trial strategy, time-locked or limited version, the firm should choose to achieve optimal result.

In particular, this paper addresses the following research issues. First, should the software firm offer only the commercial product or a commercial product together with a time-locked free trial? Second, if it is more profitable to offer a time-locked free trial, what is an optimal length of time for the free trial and what is an associated optimal price for the commercial product? Finally, under what condition will the time-locked free trial be a better strategy than the limited version free trial?

My research provides several useful insights into whether the software firm should offer a time-locked free trial. First, I find that when the network effect is not very strong and the consumers' prior expectation about the software is relatively low, the firm should introduce the time-locked free trial software. Second, when it is in the firm's best interests to introduce the time-locked free trial to the market, the optimal free trial time depends on, among several factors, the network effect and consumers' prior expectation. In general, the stronger the network effect or the higher the consumer's prior belief about the functionality of the software, the shorter the optimal free trial time. The common industry practice of thirty-day free trial period is not always optimal for all software producers. Failing to set the optimal trial time can result in substantial profit losses. Finally, the time-locked free trial strategy analyzed in this paper outperforms the limited version free trial when the network effect is moderate. If the network effect is intense, the firm should opt for a limited version free trial to exploit the positive network effect.

Literature Review

Prior studies on network externalities effects provide interesting perspectives on the free trial software problem. For example, free trial software can be considered as an inferior version of the commercial product. Conner (1995) proposes that when the network effect is intense, it is profitable for an innovator to allow its competitors to "clone" its product with lower functionality. However, when it is the same software firm that offers both the higher

functionality commercial version and the lower functionality limited version free trial, Conner's (1995) model would dictate that the firm should set the functionality of the free trial product to the lowest possible value, i.e. zero, in order to maximize its profit. To address the unique zero-price nature of the free trial software, Cheng and Tang (2007) extend Conner's (1995) models to study the optimal strategy for the limited version free trial software by explicitly considering consumers' aggregate software usage cost that includes, for example, search cost associated with finding a particular software product of interest and learning cost of time and effort spent to get familiarized with the functionalities of the software. They show that under a strong network effect a firm will introduce limited version free trial software, and the firm is better off offering a free trial limited version than segmenting the market by charging a price for the product of lower functionality. Interestingly, we find that the time-locked free trial strategy is favorable to the firm when the network effect is not intense. We show that the time-locked free trial strategy outperforms that of the limited version when the network effect drops below a certain threshold. This finding suggests that these two forms of free trial are complementary rather than competing against each other.

Product trial and sampling, a long standing research topic in marketing, is another stream of literature relevant to software free trial issue. Theoretical quantitative models of sampling behavior include the pioneering work of Jain, et al. (1995) and Heiman, et al. (2001), who model sampling as an important factor in the diffusion of new products. Jain et al. (1995) use simulation to determine an optimal level of product sampling for a new product. To do so, they modify the Bass diffusion model by assuming that the coefficient of innovation is a function of the sampling level. They find that sampling is critical in the initial stages of a product's life since increasing the number of first adopters not only leads to a future customer base but also provides a source

for product promotion by word of mouth. Their analysis gives insight into the optimal initial level of investment in sampling and how this affects the dynamics of product diffusion. Similarly, Heiman et al. (2001) develop a model that decomposes the sampling effort into the immediate sales and longer-run (goodwill-building) effects. Their model identifies an optimal sampling effort of a firm over time. More recently, Bawa and Shoemaker (2004) study the effects of free samples sent to and consumed at home via direct mail. They examine three potential effects of free direct mail samples on sales: (1) an acceleration effect, whereby consumers begin repeat purchasing of the sampled brand earlier than they otherwise would, (2) a cannibalization effect, which reduces the number of paid purchases of the brand, and (3) an expansion effect, which induces purchasing by consumers who would not consider buying the brand without a free sample. They find that free samples can produce measurable long-term effects on sales and the effectiveness of a free sample promotion can vary widely, even between brands in the same product category.

Most of the prior studies of product trial, defined as a consumer's first usage experience with a brand or product by Kempf and Smith (1998), have been conducted in a laboratory setting (e.g. having the trial taking place in a show room) and examine the impact of the trial on belief strength and attitude (Marks and Kamins, 1988), affect (Oliver 1992), or perceptions of the brand (Bettinger et al. 1979). While the abovementioned studies of product trial and sampling, using both internal (i.e., the show room product trial) and external (i.e., direct mail sampling) laboratory experiments, provide insights into consumers' cognitive and emotional response to a product, this paper addresses some unique issues of software free trial. First, regular product trial or sampling are usually available to a group of pre-selected customers, while free trial software is available for downloading to all interested consumers. As a result, software companies risk

losing more demand to the free trial users. Second, the difference between small samples and the underlying commercial product is in quantity not in functionality (e.g., small samples of Shampoos). In this respect, the length of free trial time amounts to the limit of consumption quantity of free samples. This free trial time, as I will show later, has significant impact on the firm's optimal profit, and thus should be treated as an important decision variable for the firm. Third, the marginal production cost of information goods is negligible, whereas that of the physical goods is not. Finally, most prior analysis of product trial does not take the network effect into consideration, while I show in this paper the network effect is one of the key factors for the decision of software free trial.

The research by Heiman and Muller (1996) is perhaps most relevant to our study on issues involved in time-locked software free trial. Investigating the optimal length of demonstration time for manufactures in such industries as motor vehicle and computer hardware, they find that the time needed by the consumers to learn about the performance of different features of a product varies from one feature to another; and the longer the demonstration, the more information is gained by the consumer. They prove both theoretically and experimentally that given consumers' prior expectation about the functionality of the product, the probability of purchase after the demonstration will first increase with more demonstration time and then decrease. This finding is similar to what I discover in my analyses reported in later section of Chapter 2. I find that the total profits of a software producer who offers a time-locked free trial will first increase with the length of free trial time and then decrease. However, our study aims at finding the optimal trial time that maximizes software firms' profits, while Heiman and Muller (1996) focuses on finding the optimal demonstration time that maximizes consumer's probability of purchase.

Software Firm's Free Trial Problem

In the study of demonstration time by Heiman and Muller (1996), the firm determines the time of demonstration and the number of potential customers to receive a demonstration. Software companies, however, cannot pick and choose which customers will download and try their software. Therefore the only leverage left in their hands is the length of free trial time so that the loss of demand is balanced against the benefit of consumers' increasing willingness to pay due to free trial.[1] This section describes the free trial product design problem facing the software firm. First, I treat the trial time as exogenous and derive the optimal price for the commercial product. Then I relax this assumption to find both the optimal length of the free trial and the optimal price of the commercial product. In both cases, I derive conditions under which the software firm is better off providing a time-locked free trial.

The Model

Consider the software firm selling a software product of functionality s_0 at price P. If the firm chooses to offer a free trial version, let the length of the free trial time be denoted by τ. Without loss of generality, τ is normalized between 0 and 1 by dividing the free trial time by the expected life span of the software product under consideration. Moreover, let t denote consumers' expected usage time of the software. For simplicity, t is uniformly distributed among all consumers and again normalized between 0 and 1.

Following the model setup in Conner (1995), let K be the size of the total population in market, N be the number of potential customers for the software product under consideration, and $K = aN$, where $a > 1$.[2] Then, $K - N = (a-1)N$ represents those who are not interested in the

[1] For the ease of presentation, the term free trial henceforth refers to time-locked free trial unless noted otherwise.

[2] K and a are introduced for the sake of notation completeness. They are not needed in subsequent analyses.

software. When N is normalized to 1 without loss of generality, the intervals $\left[-(a-1),1\right]$ and [0,1] represent the total population and potential customers of the market respectively. For simplicity, let each individual's "type" (i.e., preference of functionality) for the software be denoted by θ uniformly distributed over $\left[-(a-1),1\right]$, see Table 2-1 for summary of notation.

Network externality increases each consumer's taste (i.e., θ) by γQ_τ, where Q_τ is the installed base or network size of the software and γ is the network effect intensity, which reflects the increase in willingness to pay when an additional consumer joins the network. Network effect intensity is a conceptual metric for the interdependence among consumers, and different software may have different magnitudes of network effect intensity. For example, communication software (i.e. Instant Messenger, Skype) typically demonstrate stronger network effect than regular anti-virus software or most PC games. Let θ_τ denote the marginal consumer who is indifferent between adopting the software and doing without. Then, consumers to the right of θ_τ represent those who adopt the software. If a time-locked free trial version is offered, the software users can be divided into those with long term usage who will purchase the software and those with short-term usage who will enjoy a free ride with the time-locked free trial.[3] Figure 1-1 shows the distribution of potential consumers.

Shapiro (1983) points out that the optimal pricing strategies are qualitatively very different under cases when consumers initially over estimate $(s > s_0)$ or underestimate $(s < s_0)$ the functionality of a product. In a more recent study, Chellappa and Shivendu (2005) also separate the information good that have been either overestimated or underestimated. In their model,

[3] Software producers today implement various mechanisms that render it useless to repeatedly remove and re-install the time-locked free trial software, preventing consumers with long term usage needs from using the time-locked software indefinitely.

piracy (similar to free trials) offers a consumption opportunity before purchase. They develop a two-stage model of a market composed of heterogeneous consumers in their valuation and moral costs. The model allows consumers to pirate the product in the first stage and based on their experience in the prior stage update their beliefs, which may cause them to re-evaluate their buying/pirating decision in the second stage. I realized that time locked free trial strategy in essence is the legal counterpart of pirating except with no moral cost involved. Similar to the model in Chellappa and Shivendu (2005), time-locked strategy allows consumers evaluate the software product in the first stage and based on their experience re-evaluate their buying/discarding decision in the second stage. Under perfect information, consumers are fully aware the existence of the free trial product. If we assume a consumer incurs the same learning/usage cost of either trial or purchase the product, s/he always obtains higher utility by trialing the software instead of buying it in the first stage, and hence for rational consumers they will first trial the software and purchase or discard it upon the trial time expire. As a result, in this paper, I choose to focus my analysis at the beginning of the second stage where a purchase decision is being made.

Offering time-locked free trial software however becomes meaningless if consumers overestimate the functionality of the commercial software. In this case, the software firm will choose not to offer any free trial product because doing so will only "disappoint" the potential consumers and reduce their willingness to pay. Consequently, for the rest of the paper, I will turn my attention to the case where consumers underestimate the functionality of commercial software. In addition, I assume that the average consumers' prior belief is known to the software firm before the release of its software.[4] This can be achieved based on either historical data or

[4] In the absence of free trials, Bergemann and Schlag (2007) propose a regret-minimizing pricing strategy when the seller has minimal information regarding the true valuation of the buyer.

market research. Let s be the consumer's prior belief about the software functionality. Further, let the price of the commercial product be P, and let c denote the aggregate cost of time and effort spent by consumers to get familiarized with the functionality of the software. From the software firm's perspective, a consumer θ obtains the following net utility when purchasing the software after free trial:

$$U = \left(\theta + \gamma Q_\tau\right) \cdot \left(s + \tau\delta\right) - P - c \tag{2-1}$$

The effect of consumer's reduced uncertainty through free trial is reflected by the improvement of consumer's perceived functionalities about the software which rises from s before the free trial to $s + \tau\delta$ after the free trial. The speed of a consumer update his belief is given by δ so that the longer the trial time the more functionality may the user recognize. Certainly, the perceived functionality should never exceed the true functionality (s_0) offered by the commercial software, hence I restrict $s + \tau\delta < s_0$. Note that, I assume the improvement of consumer's belief is linearly proportional to the length of the free trial time. Similar assumptions are observed in Pynadath and Marsella (2004) and Liu, et al. (2003) where linear belief update functions are recommended techniques that can exploit structured problem domains to more efficiently solve the modeling and planning tasks in decision-theoretic frameworks. However, it is conceivable that consumers update their beliefs faster at the beginning of the trial as the gap between their prior expectations and true functionality is large, and the belief updating slows down as the gap diminishes. This learning process can be captured by a concave belief updating function. I show in later section that the linear function in my model offers exactly the same managerial insights to the time-locked free trial problem as does a concave function.

Recall that θ_τ denotes the marginal consumer type who is indifferent between adopting the commercial software and doing without. Setting the net utility function (Equation 2-1) to zero, one derives the marginal consumer type as:

$$\theta_\tau = \frac{P+c}{s+\tau\delta} - \gamma Q_\tau \tag{2-2}$$

Consumers of higher valuation type than the marginal consumer $(\theta > \theta_\tau)$ have positive net utility and if their expected usage time is longer than the trial time $(t \geq \tau)$, they will purchase the software.[5] Conversely, consumers with lower valuation type $(\theta < \theta_\tau)$ or consumers with expected usage time shorter than the trial time $(t < \tau)$ will not buy the software. The intuition may suggest that the software firm should decrease the free trial time to minimize short term users who enjoy a free ride. One can easily derive from Equation 2-2 that decreasing free trial time will lead to the marginal consumer's type shifting to the right, i.e., reducing the number of potential buyers, see Figure 2-1. Increasing the trial period improves the valuation by each type and a higher price can be charged by the software firm. A longer trial period, however, implies increased free riders. That is, there exists a tradeoff between lowering the marginal consumer type and decreasing the free trial time.

Since both t and θ are uniformly distributed, the area under the product of two sub-intervals $[\theta_\tau, 1] \times [\tau, 1]$ corresponds to buyers (i.e., the demand) of the commercial software. Thus, the demand D for the software is described by

$$D = (1-\theta_\tau)(1-\tau) \tag{2-3}$$

[5] Those consumer who derive positive utility without going through the belief updating process of using the free trial (i.e., $\theta s - P - c > 0$ as opposed to $\theta[s + \tau(s_t - s)] - P - c > 0$) will purchase the commercial product directly.

Substituting Equation 2-2 into 2-3 and invoking rational expectation equilibrium $D = Q_\tau$, one derives the demand (i.e., the installed base Q_τ) of the commercial software as

$$Q_\tau = \frac{(1-\tau)(s+\tau\delta-P-c)}{(s+\tau\delta)(1-\gamma+\gamma\tau)} \tag{2-4}$$

Since both $(1-\tau)$ and $(s+\tau\delta)$ are positive, $Q_\tau > 0$ implies that

$$0 < P < s+\tau\delta-c \tag{2-5}$$

$$0 < \gamma < \frac{1}{1-\tau}, \text{ and} \tag{2-6}$$

$$\tau > \max\left\{\frac{c-s}{\delta}, 0\right\} \text{ since } s+\tau\delta-c > 0 \tag{2-7}$$

Inequality 2-5 implies that the higher the true software functionality, the higher the price the software firm can charge for its commercial product. This is consistent with our previous observation on pricing the commercial product. It is worth noting that when both Inequalities 2-5 and 2-6 are the other way around, the condition of positive demand is still satisfied. However, this in turn renders a positive relationship between the price and the demand and setting $P = \infty$ leads to $Q_\tau \to \infty$ for any $\gamma > \frac{1}{1-\tau}$. Obviously, this result contradicts the law of demand where the higher the price, the lower the quantity demanded. As a result, we require the constraint for positive demand be the format as given by Inequalities 2-5 and 2-6. Inequality 2-7 shows that there is a trial time threshold, which is decided by the aggregate cost and consumer's prior belief. When $c < s$, τ can be any number between 0 and 1. If $c > s$, which implies the cost of using the software is higher than consumer's prior expectation about the functionality of the software, then $\frac{c-s}{\delta}$ is the least amount of free trial time a firm should offer if they want to secure positive demand. In reality, sophisticated software turns to scare potential customers away as it always

demand higher learning cost. Offering trial time for a longer time helps reduce the "pain" of learning and I have observed that large companies have shown the effort of pushing the time-locked strategy to its limit in an aggressive way as the software become increasingly complicated. For example, Microsoft starts offering its newly released Visual Studio 2008, Internet Security and Acceleration Server 2006, Windows Server 2008 for 90 days, 180 days and 240 days free trial respectively. These software are often considered by many to be sophisticated and often incur higher learning cost to get acquaint with.

To determine the impact of a change in γ on Q_τ, while holding τ constant, one has the following:

$$\frac{\partial Q_\tau}{\partial \gamma} = \frac{(1-\tau)^2(s+\tau\delta-P-c)}{(s+\tau\delta)(1-\gamma+\gamma\tau)^2}, \text{ and} \tag{2-8}$$

$$\frac{\partial Q_\tau}{\partial \gamma} > 0 \text{ if } (s+\tau\delta-P-c) > 0, \text{ or equivalently } \tau > \frac{P+c-s}{\delta} \tag{2-9}$$

Inequality 2-9 implies that the increase of network effect will increase the number of total buyers if the length of the free trial is greater than $\frac{P+c-s}{\delta}$. Similarly, to determine the impact of a change in τ on Q_τ, while holding γ in constant, we have:

$$\frac{\partial Q_\tau}{\partial \tau} = \frac{(s+\delta(1-\gamma(1-\tau)^2))(P+c)-(s+\tau\delta)^2}{\left(1-\gamma+\gamma\tau\right)^2\left(s+\tau\delta\right)^2}$$

Depending on the underlying parameters, the impact of a change in the free trial time on the total demand of the software could be first positive and then negative or negative all the time as the free trial time τ varies from 0 to 1. But for most of the times the total demand will first increase at the beginning of the trial, but decreases and eventually converges to zero as the free trial being offered for a longer time. Indeed, offering a trial time that is shorter than optimal

reduces the free riders, but doing so also reduces the consumer's willingness to pay which in turn weakens the demand of the software. On the other hand, offering a trial time that is longer than optimal increases consumer's willingness to pay but increases the number of free riders too. One of the major contributions of this study is that I find optimal strategies for the software firm when such a dilemma appears.

Optimal Pricing Strategy While τ is Given

In this subsection, I explore an optimal pricing strategy when the software firm faces a common established free trial period in the industry.[6] When the increment of consumer's belief (δ), the free trial period (τ), the consumer's prior belief (s), the network effect (γ) and the aggregate software usage cost (c) are exogenously given, the software firm seeks to set the price of the commercial product to maximize its profit as follows:

$$\max_{P} \pi = PQ_{\tau} = \frac{(1-\tau)(s+\tau\delta-P-c)}{(s+\tau\delta)(1-\gamma+\gamma\tau)}P \tag{2-10}$$

$$\text{s.t. } 0 \le P \le s+\tau\delta-c$$

Note that the total revenue function in Equation 2-10 amounts to the profit function due to the negligible marginal production cost of software. Solving the above problem yields the following optimal price, demand and profit of the commercial software.

$$P^* = \frac{s+\tau\delta-c}{2} \tag{2-11}$$

$$Q_{\tau}^* = \frac{(1-\tau)(s+\tau\delta-c)}{2(s+\tau\delta)(1-\gamma+\gamma\tau)} \tag{2-12}$$

$$\pi^* = \frac{(1-\tau)(s+\tau\delta-c)^2}{4(s+\tau\delta)(1-\gamma+\gamma\tau)} \tag{2-13}$$

[6] For example, thirty days free trial time seems to be used by most software producers.

It can be easily shown that the optimal price, demand and overall profit for the software firm when no free trial is offered are given by

$$P^* = \frac{s-c}{2},\ Q^* = \frac{s-c}{2s(1-\gamma)}, \text{ and } \pi^* = \frac{(s-c)^2}{4s(1-\gamma)}$$

Using the above results as the benchmark and comparing them against Equations 2-11, 2-12 and 2-13, we have the following condition under which the firm is better off offering a time-locked free trial when the free trial time τ is given.

Proposition 2-1: *When the free trial time τ is given, the firm will realize more profit by offering the time-locked free trial, if the network effect γ is less than $\overline{\gamma}_1$ and $\overline{\gamma}_1 > 0$ where*

$$\overline{\gamma}_1 = \frac{(s-c)^2(s+\tau\delta)-(1-\tau)(s+\tau\delta-c)^2 s}{(1-\tau)\left[(s-c)^2(s+\tau\delta)-s(s+\tau\delta-c)\right]} \tag{2-14}$$

It can be shown that the denominator of $\overline{\gamma}_1$ in Proposition 2-1 is always positive. The numerator, however, can become negative depending on the underlying parameters s, s_t, c, and τ. This implies that introducing a time-locked free trial is not feasible when $\overline{\gamma}_1$ is negative since software products exhibit positive network effect (i.e., $\gamma > 0$) and the condition $\gamma < \overline{\gamma}_1$ is impossible to satisfy.

Finding Both Optimal Price *P and Optimal Trial Time *τ****

By relaxing the restriction that the free trial time is exogenously given, the problem in the previous section is generalized into the problem where the software firm seeks to find both an optimal free trial time and optimal price of its commercial product so that its profit is maximized as follows:

$$\max_{\tau,\,P} \pi = PQ_\tau = \frac{(1-\tau)(s+\tau\delta-P-c)}{(s+\tau\delta)(1-\gamma+\gamma\tau)}P$$

s.t. $0 \le P \le s + \tau\delta - c$, and $0 \le \tau \le 1$

To solve this problem, we first treat the free trial time τ as given, and the problem reduces to the case in the previous section where the optimal price and the overall profit as a function of τ are $P^*(\tau) = \dfrac{s + \tau\delta - c}{2}$ and $\pi^*(\tau) = \dfrac{(1-\tau)(s+\tau\delta - c)^2}{4(s+\tau\delta)(1-\gamma+\gamma\tau)}$ respectively. Then, we evaluate the change of $\pi^*(\tau)$ with respect to τ by having τ ranges between 0 and 1 as below:

$$\max_{\tau} \pi^*(\tau) = \frac{(1-\tau)(s+\tau\delta - c)^2}{4(s+\tau\delta)(1-\gamma+\gamma\tau)} \tag{2-15}$$

s.t. $0 \le \tau \le 1$

One observes from the objective function that setting free trial time equal to the expected life span of the software, i.e., $\tau = 1$, leads to zero profit and it is quite clear that this is not an optimal solution to the software firm's maximization problem. Therefore, the constraint $\tau \le 1$ is not binding. Solving the above maximization problem given that $\tau^* \ne 1$ leads to

$$\tau^* = \begin{cases} \sqrt[3]{-\frac{u}{2} - \sqrt{\left(\frac{u}{2}\right)^2 + \left(\frac{v}{3}\right)^3}} + \sqrt[3]{-\frac{u}{2} + \sqrt{\left(\frac{u}{2}\right)^2 + \left(\frac{v}{3}\right)^3}} & \text{if } 0 \le \gamma \le 1 - \frac{s(s-c)}{\delta(s+c)} \\ 0 \quad \text{(i.e., no free trial)} & \text{Otherwise} \end{cases} \tag{2-16}$$

where:

$$u = \frac{2\left(c^3\gamma^3 + 3c\gamma\left(s\gamma - 4\delta + \gamma\delta\right)\left(s\gamma - \delta + \gamma\delta\right) + 3c^2\gamma^2\left(s\gamma + 2\delta + \gamma\delta\right) + \left(s\gamma - \delta + \gamma\delta\right)^2\left(s\gamma + 8\delta + \gamma\delta\right)\right)}{27\gamma^3\delta^3}$$

$$v = -\frac{\left(s+c\right)^2\gamma^2 + \gamma\delta\left(4c - 5s + 2\left(s+c\right)\gamma\right) + \left(4 - 5\gamma + \gamma^2\right)\delta^2}{3\gamma^2\delta^2}$$

Following the above results, we derive the condition under which the firm will introduce the time-locked free trial when both the price of its commercial software and the free trial time length have to be decided:

Proposition 2-2: *Define* $\overline{\gamma}_2 = 1 - \frac{s(s-c)}{\delta(s+c)}$. *The software firm prefers to offer the time-locked free trial for greater profit if* $0 < \gamma < \overline{\gamma}_2$. *The optimal trial time* τ^* *is specified in Equation 2-16. The firm sets the optimal price at* $P^* = \frac{s + \tau^*\delta - c}{2}$, *and realizes optimal profit* $\pi^* = \frac{(1-\tau^*)(s+\tau^*\delta - c)^2}{4(s+\tau^*\delta)(1-\gamma+\gamma\tau^*)}$. *Introducing the time-locked free trial, however, is not desirable if* $\gamma \geq \overline{\gamma}_2$.

Proposition 2-2 states that the software firm will realize more profit by introducing the time-locked free trial if the network effect is below the $\overline{\gamma}_2$ threshold. To illustrate Proposition 2-2, we plot the behavior of optimal trial time (τ^*) and the maximized profit (π^*) with respect to various values of network effect intensity.

The two parameters we use in the following numerical analysis are $s \in [10\%, 60\%]$, $c = 10\%$ of the true functionalities offered by the software. The scale and ratio of the parameters are chosen to reflect the relative gap between different variables in the model. For example, we may think of consumers have perceived 60% of the true functionalities before trial and the incurred aggregate learning/usage cost is proportional to the complexity (functionalities offered) of a software. Choosing $c = 10\%$ corresponds to a low aggregate cost so that the effect of c will not overshadow the effect of free trial. Note that we start plotting consumer's prior belief from $s = 10\%$ since it is required to have $s - c > 0$ when $\tau = 0$ in order to ensure a positive demand. Also for consumers with prior belief $s > 60\%$, as suggested by proposition 2-2, it is no longer

optimal to offer any kind of time-locked free trial, we therefore set up the upper bound for *s* equals to 60%.

In Figure 2-2(A), the shape of the overall profit curve shifts from concave to convex as the network effect increases and the point where the overall profit function changes from concave to convex is the threshold value $\overline{\gamma}_2$. Two observations are in order regarding Figure 2-2. First, the overall profit increases with γ due to the positive network effect. Second, the optimal trial time τ^* decreases (as shown by Figure 2-2(B)) as the network effect increases and eventually converges to zero (i.e., no free trial is desirable) when the network effect exceeds the threshold value $\overline{\gamma}_2$.

Another interesting finding from Proposition 2-2 is that the threshold value $\overline{\gamma}_2$ decreases with the increase of consumer's prior belief. This implies that the higher the consumer's prior belief, the less likely the time-locked free trial strategy is desirable. Recall that the best strategy for the software firm is to not offer any free trial if consumers initially overestimate the software. Apparently, even if consumers initially underestimate the functionality about a software product, the vendor may still be better off in the presence of a high consumers' prior belief. To illustrate this insight, Figure 2-3 shows the threshold value $\overline{\gamma}_2$ as a decreasing function of consumer's prior belief *s*.

Why Setting an Optimal Free Trial Time Matters

From the perspective of software vendors, a question of significant interests is whether it matters to find an optimal length of free trial time. A thirty-day free trial seems to be an intuitively reasonable time. Gallaugher and Wang (1999) have shown that firms that offer a thirty-day trial version of their products were able to price their products roughly 1.33 times higher than were firms that did not offer a trial, but whether the 30-day trial strategy generate

more profit remained uncertain. Investing in market research to set up an optimal trial time is desirable only if doing so brings in more profit to the vendors. We exhaust all possible combinations to evaluate the performance of a thirty-day free trial and an optimally set trial time with various consumers' prior beliefs and demonstrate through an example in Figure 2-4A and 2-4B that the profit difference between the two can be substantial, especially when the network effect of the software is low.

In Figures 2-4A and 2-4B, we explore the behavior of profits of no free trial, a thirty-day free trial, and a free trial with an optimal trial time with respect to the changes of network effect and consumer's prior belief. As shown in Figure 2-4A, when consumer's prior belief is fixed, a thirty-day fixed time free trial strategy performs only marginally better than the no free trial strategy. However, once the firm implements the optimal trial time, a significant improvement in profit is observed. In the shown experiment, the profit gain of adopting an optimal trial time over a thirty-day free trial can be as much as eighty percent, which provide the much needed incentives for software vendors to setup an optimal trial time according to the one specified in Equation 2-16.

In Figure 2-4B, we allow both consumer's prior belief and the network effect to change. The same finding is observed. The surface of setting an optimal free trial time dominates that of the thirty-day free trial, and the lower the consumer's prior belief, the larger the profit gap between the two. We notice that recently Microsoft offers a full year free trial rather than the traditional thirty-day evaluation for its Windows 7 operating systems. As software becomes more sophisticated, the gap between consumers' prior belief and the software's true functionality will become larger, leading to a longer optimal trial time.

Discussions and Further Insights

Correlation Between Consumer Type and Usage Time

The consumers in our model are uniformly distributed over consumer type θ and usage time t in Figure 2-1. This uniform distribution implicitly implies an independence between θ and t. The area under the product of two sub-intervals $[\theta_\tau, 1]\times[\tau, 1]$ therefore corresponds to the demand (i.e. buyers) of the software. However, consumer's high valuation for functionality may be caused by the need to use the software for a long period of time. That is, there may exist a positive correlation between consumers' expected software usage time and their valuation type.

To account for the positive correlation between consumer's type and the software usage time, I let the consumer's expected usage time increase linearly with his valuation type. As a result, for any potential buyer with his type between $[\theta_\tau, 1+\gamma Q_\tau]$, there are now more consumers with their usage time longer than the trial time $(t > \tau)$. Figure 2-5 presents the extended model incorporating a linear positive correlation between consumer type and expected usage time, where the lower triangle area *A* represents the added buyers due to this positive correlation.

In Figure 2-5, the proportion of the added buys increases from 0 to $2\rho\tau(1-\tau)$ as θ increases from θ_τ to 1 due to the positive correlation. The intensity of the correlation is captured by the parameter ρ, where $0 \le \rho \le 1$ is the correlation between t and θ. A higher value of ρ signifies a stronger correlation between t and θ. The total number of buyers in the positive correlation case is thus equal to:

$$D = (1-\theta_\tau)(1-\tau) + \rho\tau(1-\tau)(1-\theta_\tau) \tag{2-17}$$

We note that the new demand function must meet the following three requirements.

1. When $\tau = 0$, the model should converge to the no free trial case where the demand of the commercial software equals to $1-\theta_\tau$.
2. When $\tau = 1$, since the trial time is unlimited, all users enjoy the free ride and hence the demand of the commercial software should equal 0.
3. As the free trial time τ varies in $[0,1]$, the total number of buyers should never exceed that of the market potential: $(1-\theta_\tau)\times 1$.

It can be easily verified that the demand function of Equation 2-17 satisfies all three conditions.

Invoking the rational expectation equilibrium by setting $D = Q_\tau$, the demand of the commercial software is found as:

$$Q_\tau = \frac{(1-\tau)(1+\rho\tau)(s+\tau\delta-P-c)}{(s+\tau\delta)(1-\gamma(1-\tau)(1+\rho\tau))} \tag{2-18}$$

Plugging the new demand Q_τ into the same maximization problem as shown in Equation 2-9, one derives the optimal price and profit for the software firm when the free trial time is exogenously given as follows.

$$P^* = \frac{s+\tau\delta-c}{2} \tag{2-19}$$

$$\pi^* = \frac{(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2}{4(s+\tau\delta)(1-\gamma(1-\tau)(1+\rho\tau))} \tag{2-20}$$

Compare the above results to Equation 2-11 and 2-12, I find that the optimal price remains the same. The optimal profit, not surprisingly, increases due to the positive correlation between consumer type and software usage time. Similar to Proposition 2-1, I derive the conditions under which the firm is better off offering time-locked free trial in the presence of the above mentioned positive correlation when the free trial time τ is given:

Proposition 2-3: *Under positive correlation between t and θ, when the free trial time τ is given, the firm will realize more profit by offering the time-locked free trial if the network effect γ is less than $\overline{\gamma}_3$ and $\overline{\gamma}_3 > 0$ where:*

$$\overline{\gamma}_3 = \frac{(s-c)^2(s+\tau\delta)-(1-\tau)\left(1+\rho\tau\right)(s+\tau\delta-c)^2 s}{(1-\tau)\left(1+\rho\tau\right)\left((s-c)^2(s+\tau\delta)-(s+\tau\delta-c)^2 s\right)} \tag{2-21}$$

Threshold values as defined in Proposition 2-1 and 2-3 correspond to the attractiveness of the time-locked free trial strategy to the software firm. Comparing the two threshold values derived with (e.g., $\overline{\gamma}_3$) or without (e.g., $\overline{\gamma}_1$) the presence of positive correlation, we find that

$$\overline{\gamma}_3 - \overline{\gamma}_1 = \frac{\rho\left(s-c\right)^2\left(s+\tau\delta\right)}{\delta\left(1-\tau\right)\left(1+\rho\tau\right)\left(\left(s^2-c^2\right)+s\tau\delta\right)} \geq 0 \tag{2-22}$$

Inequality 2-22 implies that time-locked free trial strategy becomes more attractive to the software firms when there is a positive correlation between consumer type and software usage time. To illustrate this point, we let $\rho = 1$ to model the strongest correlation and suppose that the life span of the software under consideration is two years (before it is replaced by a major upgrade or a new version). Assume the software firm follows the common 30-day free trial time strategy. Normalizing the free trial time with respect to the life span of the software, we have $\tau = \frac{30}{365 \times 2} = 0.041$. Further, we let $s = 6, \delta = 4, c = 2$, indicating that users' prior belief is 60% of the true functionality of the software and the learning cost is relatively small, resulting $\overline{\gamma}_1 = 0.22$ and $\overline{\gamma}_3 = 0.97$. If the network effect of this software is between 0 and 1, then according to Proposition 2-3, the software firm will find it is more profitable 97% of the time to offer the 30-day free trial when there exists a positive correlation between t and θ. The same percentile drops to only 22% when this correlation is ignored.

Proposition 2-4 specifies a new threshold value of network effect $\overline{\gamma}_4$ when the software firm optimally sets both the price and the free trial time in the presence of a positive correlation between consumer type and software usage time.

Proposition 2-4: *Define* $\overline{\gamma}_4 = \dfrac{(s+\delta)(s+c)-2s^2+\rho s(s-c)}{\delta(s+c)}$. *Under positive correlation between* t *and* θ, *the software firm prefers to offer the time-locked free trial for greater profit if* $0<\gamma<\overline{\gamma}_4$. *Introducing the time-locked free trial, however, is not desirable if* $\gamma \geq \overline{\gamma}_4$.

Once again, we find $\overline{\gamma}_4 \geq \overline{\gamma}_2$ for any value of ρ, indicating that time-locked strategy outperforms that of the no free trial policy for a wider range of network effects. Combining the results from Proposition 3 and 4, it becomes apparent that it is more favorable for the software firm to offer time-locked free trial when there exists a positive correlation between t and θ.

Optimal Trial Time under Correlation of Consumer Type and Software Usage Time

How does the optimal trial time vary when the correlation between software usage time t and consumer type θ increase? Intuitively, the stronger the correlation between t and θ, the less likely a consumer of type θ will be a short term user. The total number of short term users under strong correlation is therefore much less than that of the no correlation case analyzed in previous sections. Hence, the software firm can take advantage of this correlation by offering longer trial time to increase consumer's willingness to pay with lessened degree of demand cannibalization. Figure 2-6 shows that the optimal trial time raises from $\tau = 0.033$ to $\tau = 0.343$ as the correlation ρ increases from 0 to 1. In another word, if the life span of a software is two years, then an optimal trial time could be as long as eight months under the strongest correlation between t and θ.

Propositions 2-1 and 2-2 indicate that the optimal trial time will eventually converge to 0 as the network effect increases. After then, the software firm is better off by choosing not to offer the free trial product. Figure 2-7 illustrates that a positive correlation between t and θ helps delay the converging process of the optimal trial time and renders the time-locked strategy more attractive to the firm for a wider range of γ. For example, the time-locked free trial is preferred by the software firm with $\gamma \in [0, 0.25]$ if t and θ are independent. The same interval expands to $\gamma \in [0, 0.45]$ when the intensity of correlation increases to $\rho = 0.25$ in Figure 2-7.

Concave or Linear Belief Updating Function

While a concave belief updating function might be more intuitively appealing, I demonstrate via computational analyses in this section that a linear belief updating function generates the same profit behavior pattern as a concave one, see Figures 2-8A and 2-8B. In the computational analyses, I replace the linear belief updating function τ in consumers' utility function by a concave function $\sqrt{\tau}$.

Notice from Figure 2-8B that the curve of the overall profit is shifted leftward with a concave belief updating function and the maximum profit is achieved with a smaller length of free trial time. Further, this concave belief updating function allows the software firm to realize more profit than a linear one. This results from the fact that, in the concave belief updating function model, consumers require less free trial time to realize the true functionality about the software and therefore the firm surrenders less demand to the short term users who enjoy free ride.

A Unified Framework for Limited Version and Time Lock Free Trial

Limited Version Free Trial

Instead of offering a time-locked free trial, the software firm has the strategic option of providing limited version free trial software to the market. As shown in Figure 2-9, there are two marginal consumers in the model of limited version free trial – the marginal consumer indifferent between doing without and trying the free software (denoted by θ_L) and the marginal consumer indifferent between the free and the commercial software (denoted by θ_H). I consider the rational expectation equilibrium where consumers do not switch between products and there are no repeat purchases. Further, each individual in the market chooses one and only one of the following three options: do without, use the free trial software, or buy the commercial software. A potential consumer of type θ derives $(\theta+\gamma Q)\cdot s_L - c$ net utility from trying the software and $(\theta+\gamma Q)\hat{s} - P - c$ from purchasing the software, where s_L is the functionalities offered by the trial product and $\hat{s}$ is consumers' belief about the functionalities offered by the commercial software, on which they base their purchase decisions. The market segment between θ_L and θ_H corresponds to the users of free trial software (Q_L), and the segment between θ_H and 1 corresponds to the buyers of the commercial software (Q_H). Since limited version free trial users have unlimited trial time, the total installed base Q of the software includes both commercial and trial users. (e.g. $Q = Q_L + Q_H$).

In order to have $Q_L > 0$, the seller has to set the price of its commercial software such that $P \geq c\frac{\hat{s}-s_L}{s_L}$. In other words, when the price of the commercial product is too low, no one wants to use the less functional trial software. All potential consumers with positive net utility will

simply buy the commercial product, which nullifies the firm's strategic offering of the free trial. Similarly, to have non-negative demand of the commercial product, the price P must satisfy an upper bound constraint: $P_H \le \frac{\hat{s}-s_L}{s_L(1-\gamma)}(s_L - c\gamma)$.

Solving the optimization problem, where the software firm seeks to set the price of the commercial product to maximize its profit, yields an optimal price and maximized profit such that: $P^* = \frac{(\hat{s}-s_L)(s_L-c\gamma)}{2s_L(1-\gamma)}$ and $\pi^* = \frac{(\hat{s}-s_L)(s_L-c\gamma)^2}{4s_L^2(1-\gamma)^2}$. It can be verified that introducing a limited version free trial is not desirable for the software firm if $\gamma < 2 - \frac{s_L}{c}$. Apparently, this threshold value decreases for a smaller software usage cost c. This explains why the software industry strives to improve the user-friendliness of the software product, which lowers consumer's learning cost for using the software.

Optimal Product Design of Limited Version Free Trial

In the previous section, I treat the functionality of the free trial software s_L as an exogenous parameter. In this section, I relax this assumption to see how the optimal solution varies. The monopoly's optimal design problem is thus described as the constrained profit maximization problem as below:

$$\max_{P,s_L} \pi = PQ_H = \frac{P(s_L - c\gamma)}{s_L(1-\gamma)} - \frac{P^2}{\hat{s}-s_L}$$

$$\text{s.t. } \frac{c}{s_L}(\hat{s} - s_L) \le P \le \frac{s_L - c\gamma}{s_L(1-\gamma)}(s - s_L) \text{ and } 0 \le s_L \le s$$

The optimal functionality and price are summarized in Proposition 2-5:

Proposition 2-5. *The optimal solution to the software firm's optimal limited version free trial design problem is described by*

$$P^* = \frac{4\hat{s} + 5c\gamma - 3\sqrt{(c\gamma)^2 + 8c\gamma s}}{8(1-\gamma)}, \text{ and}$$

$$s_L^* = \frac{-c\gamma + \sqrt{(c\gamma)^2 + 8c\gamma\hat{s}}}{2}$$

The Optimal profit equals

$$\pi^*_{Limited} = \frac{8c^3\gamma^3 + 26c\gamma\hat{s}^2 + 47c^2\gamma^2 s - \left(2s^2 + 17c\gamma s + 8c^2\gamma^2\right)\sqrt{(c\gamma)^2 + 8c\gamma s}}{4(1-\gamma)^2\left(c\gamma - \sqrt{(c\gamma)^2 + 8c\gamma\hat{s}}\right)\left(c\gamma + 2s - \sqrt{(c\gamma)^2 + 8c\gamma s}\right)},$$

and the software firm prefers to offer the limited-version free trial for greater profit if

$$\frac{2c}{c+\hat{s}} < \gamma < \frac{c^2 - 4c\hat{s} + 4s^2}{c^2}.$$

I show in the following that with a simple extension, the above results can be incorporated in a unified framework and compared against optimal solutions for time-locked free trial problem derived in the previous two sections.

Similar to what I proposed in the time-locked free trial analysis, I extend the one dimensional limited version model by introducing the free trial time parameter τ. Instead of treating τ as a decision variable, we set $\tau = 1$ representing an unlimited trial time. Also similar to the time-locked free trial, limited version strategy also helps consumers reduce their uncertainties and update their beliefs about the functionalities of the commercial product. However, since trial users only have limited access to the true functionalities, we assume limited version trial users have a slower belief updating speed than time-lock users (i.e., $\delta' < \delta$). In addition, we let $\hat{s} = s + \tau\delta'$ so that the limited version trial users share the same prior belief s with the time-lock users but with unlimited trial time ($\tau = 1$) and slower belief updating speed ($\delta' < \delta$). Then, the consumer's utility function for the limited version trial users can be rewritten as

$\left(\theta+\gamma Q\right)(s+\tau\delta')-c$, in the same way as the one for the time-lock users. After this transformation, optimal solutions to the time-locked free trial problem become directly comparable to those from the limited version analysis.

Limited Version or Time Lock?

The software firm will adopt time-locked free trial strategy over limited version if the firm realizes more profit with the time-locked free trial software, i.e., $\pi^*_{Time-Lock} \geq \pi^*_{Limited}$, where

$$\pi^*_{Time-Lock} = \frac{(1-\tau^*)\left(1+\rho\tau^*\right)(s+\sqrt{\tau^*}\delta-c)^2}{4(s+\sqrt{\tau^*}\delta)\left(1-\gamma\left(1-\tau^*\right)\left(1+\rho\tau^*\right)\right)} \tag{2-23}$$

$$\pi^*_{Limited} = \frac{8c^3\gamma^3+26c\gamma\hat{s}^2+47c^2\gamma^2 s-\left(2s^2+17c\gamma s+8c^2\gamma^2\right)\sqrt{(c\gamma)^2+8c\gamma s}}{4\left(1-\gamma\right)^2\left(c\gamma-\sqrt{(c\gamma)^2+8c\gamma\hat{s}}\right)\left(c\gamma+2s-\sqrt{(c\gamma)^2+8c\gamma s}\right)}, \text{ and} \tag{2-24}$$

$$\frac{2c}{c+\hat{s}} \leq \gamma \leq \min\left\{\overline{\gamma}_4,\ \frac{c^2-4c\hat{s}+4s^2}{c^2}\right\} \tag{2-25}$$

Inequality 2-25 ensures that adopting either free trial strategy is desirable for the software firm.

Although I have closed form solutions of maximum profit to both time-locked and limited version free trial problems specified in Equation 2-23 and 2-24, it is analytically intractable to examine which strategy generates more profit for the software firm. I thus turn to numerical analyses to answer this question. Figure 2-10 and 2-11 show various maximized profits the software firm may realize using each strategy under different network effect intensity values and consumers' prior beliefs. In order to ensure that the firm has incentive to adopt both free trial strategies, I graph the profit curve only within the selected range of γ that satisfies Inequality 2-25 in the analyses.

Using the same parameters as in the previous section, Figure 2-10 compares the two free trial strategies when a concave belief updating function is utilized and a correlation of $\rho = 0.5$

between t and θ is considered for the time-locked policy. It is not clear yet how fast a consumer update his belief with a limited version trial (δ') compare to a time locked trial (δ), therefore we vary the relative gap between the two variables and plot the scenarios where δ' is 10%, 25% and 50% of δ in the shown experiment.

As shown in Figure 2-10, the trajectory pattern of overall profit is increasing in the network effect for both strategies, suggesting that a stronger network effect is beneficial for the software firm. Limited version free trial dominates the time-locked strategy by generating more profit when the its trial users has a faster belief updating speed (i.e., $\delta' = 50\%\delta$), but being dominated when this speed is slow (i.e., $\delta' = 10\%\delta$). If limited version trial users update their beliefs in a moderate pace (i.e., $\delta' = 25\%\delta$), then the limited version strategy becomes more desirable under strong network effect, while time-locked free trial is preferred when the network effect is low. For example, $\gamma < 0.56$ is the condition under which time-locked free trial strategy outperforms that of limited version when users perceive 30% of the true functionalities before the trial.

Increasing the value of consumer's prior belief s gradually shifts the trajectory pattern of limited-version free trial upward, which in turn makes it superior to the time-locked policy as evidenced by the diminished range of network effects where time-locked policy dominates that of the limited version. Using the same parameters, Figure 2-11 compares the two free trial strategies when users perceive 40% of the true functionalities before the trial. It clearly shows that the time-locked free trial is more desirable than the limited version for a smaller range of network effect intensities ($\gamma < 0.48$ compared with $\gamma < 0.56$ in the case where consumers perceive 30% of the true functionalities). A shrinking profit gap between the two strategies is also observed while increasing the value of s. Indeed, there isn't that much space for the trial

strategy to improve if consumers have already realized the true functionalities of the software to a high extend.

In sum, I find that the software firm should offer time-locked free trial over limited version free trial for greater profit if the network effect is moderate and consumer's prior belief is low. When the network effect is strong, the firm will achieve more profit by offering limited version free trial to exploit the network effect.

Summary

Free trial software, a prevalent practice for the software industry, usually comes with two flavors – limited version and a time-locked version. The limited version free trial software has some functions disabled (e.g., print, save,…, etc), but it can be used without any time constraints. The time-locked free trial allows users to access full functionality of the software for a pre-determined time, typically thirty days.

In Chapter 2, I address issues pertinent to time-locked free trial software with the following key findings. First, I find that there exists a threshold of network effect. Only when the network effect is smaller than this threshold will time-locked free trial become more profitable for the software firm than the benchmark case of no time-locked free trial. This threshold value decreases with increases in consumers' prior beliefs about the software's functionality. Second, the optimal trial time depends on a variety of factors, including the functionality of the software, consumers' prior belief about this functionality, the network effect, and the aggregate cost of using the software, and is shown to decrease as the network effect increases. Failure to set an optimal trial time and to simply conform to the common practice of a thirty-day free trial could result in substantial profit loss. Third, the existence of a positive correlation between consumers' preference for functionality and the software usage time renders the time-locked free trial strategy more attractive to the firm for a wider range of network effects. Fourth, when consumers

learn the true functionality of the software through free trial at a faster pace (e.g., when the belief updating function is concave), the firm will find the time-locked free trial strategy more favorable for a wider range of network effect. Finally, I build a unified framework to help the software firm decide which form of free trial, limited version or time-locked, is the best free trial strategy. I find that limited version free trial generates more profit in the presence of strong network effect, while time-locked free trial outperforms limited version when the network effect is modest.

There are several possible extensions to this paper. First, it is of interest to examine the impact of competition on the firm's free trial decision in a duopoly market where both firms offer substitute software products, and one firm has a functionality advantage over the other. In the presence of competition, what will be the conditions for both firms to offer time-locked free trial and what will be the best strategy to the firm that has functionality advantage? Another possible direction of future research is to examine the mixed free trial strategy by offering both limited version and time-locked free trials, a strategy albeit rarely observed so far. Several intriguing issues arise. Should the firm introduce both versions of free trials at the same time? Or, should the firm first introduce the time-locked free trial, which turns into limited version upon expiration of the free trial period? What will be the optimal trial time and what will be the optimal amount of functionalities offered by the limited version strategy in this context?

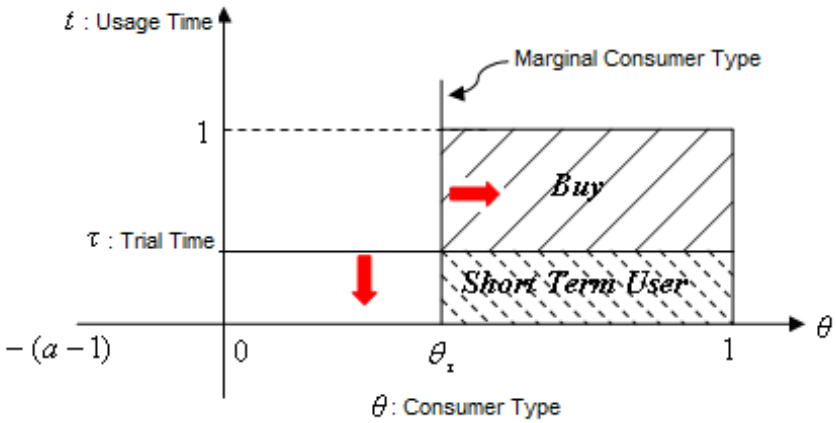


Figure 2-1. Distribution of potential consumers with network effect

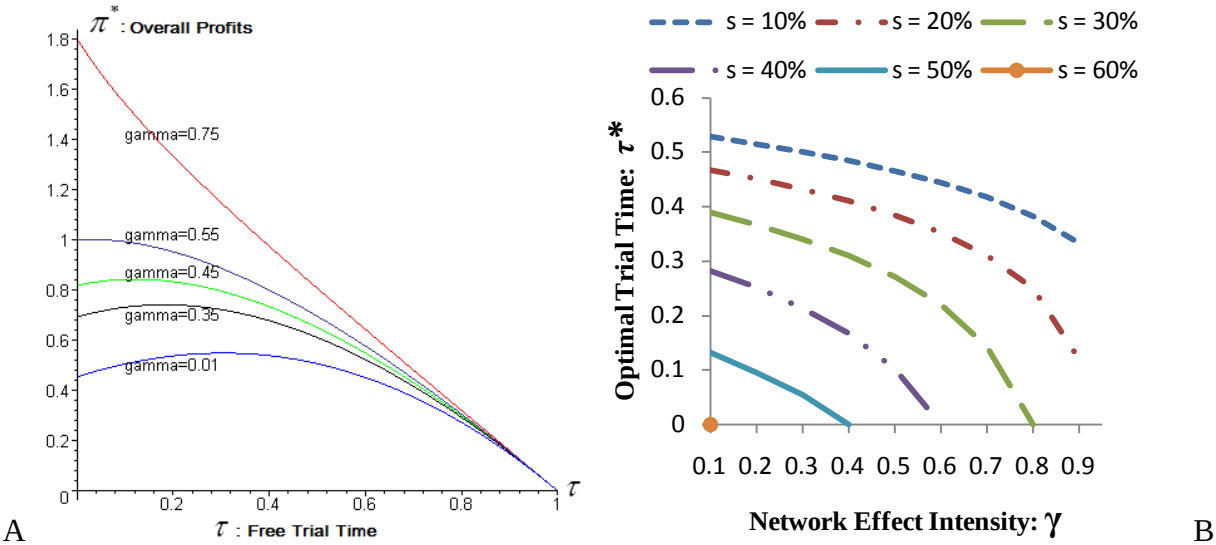


Figure 2-2. Numerical plot. A) Behavior of $\pi^*(\tau)$ with respect to τ for different network intensities. B) Behavior of optimal trial time τ^* for different network effect intensities

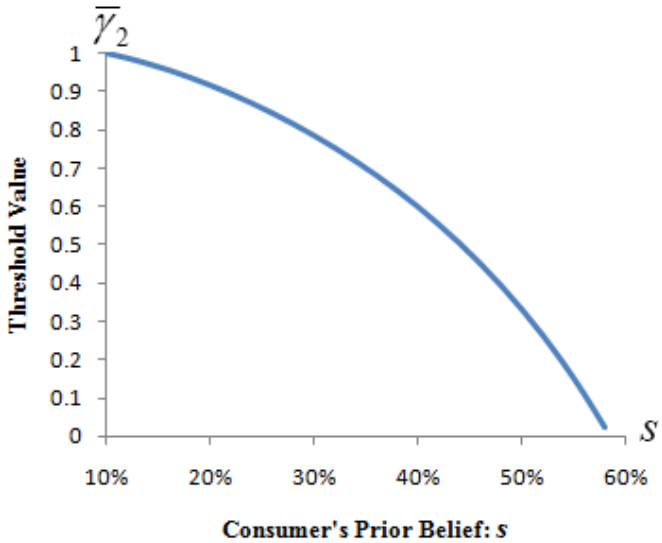
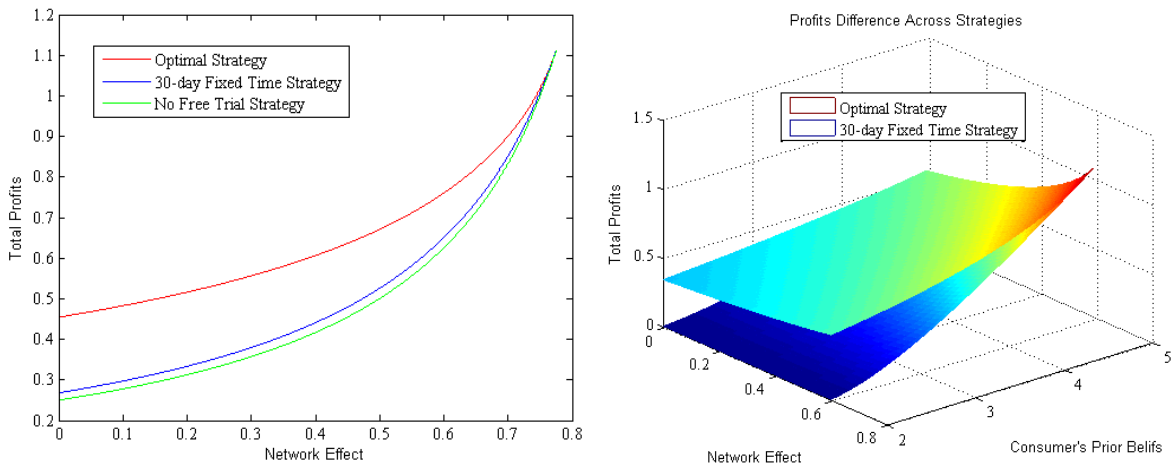


Figure 2-3. Threshold value $\overline{\gamma}_2$ vs. prior belief s



A B

Figure 2-4. Optimal strategy vs. 30-day fixed time strategy. A) Vary network effect. B) Vary both network effect and consumer's prior belief.

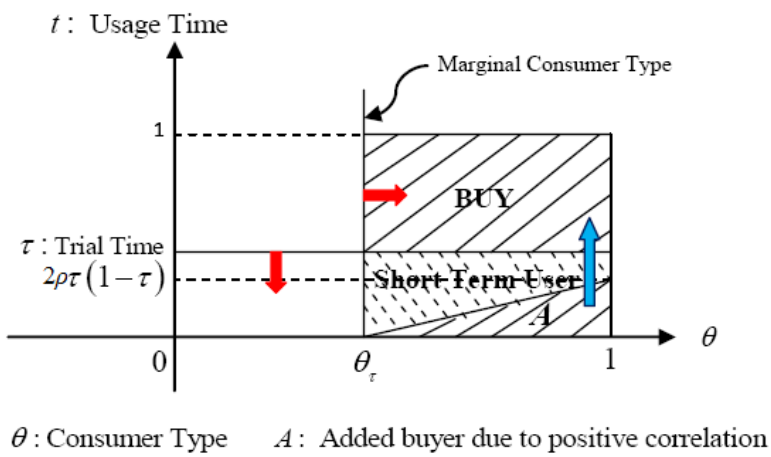


Figure 2-5. Distribution of potential consumers under positive correlation

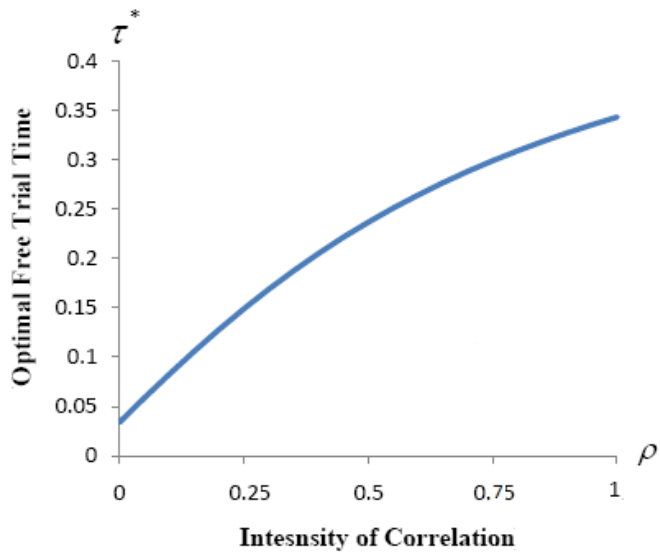


Figure 2-6. Optimal trial time under correlation

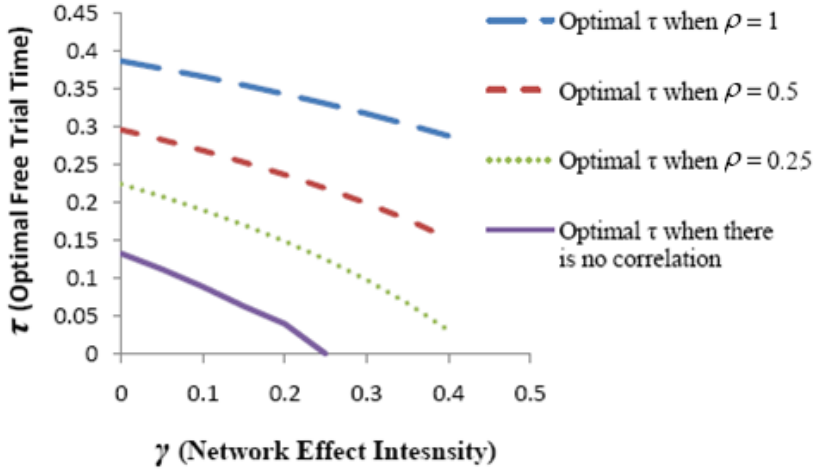
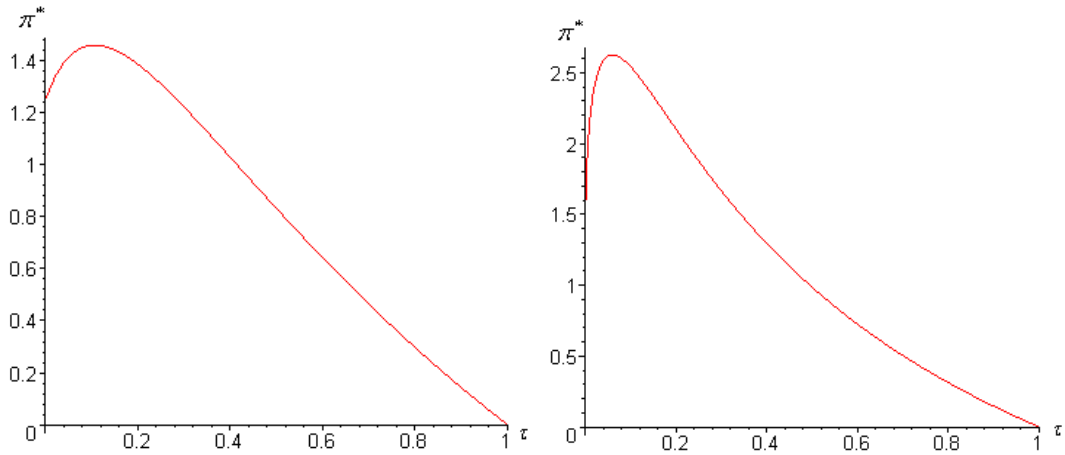


Figure 2-7. Delay of the converging speed



A B

Figure 2-8. Concave vs. linear belief updating function. A) Linear B) Concave.

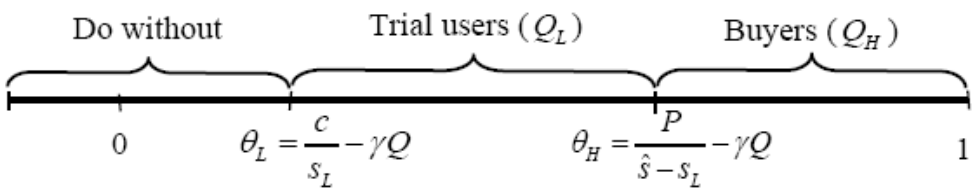


Figure 2-9. Model of limited version free trial

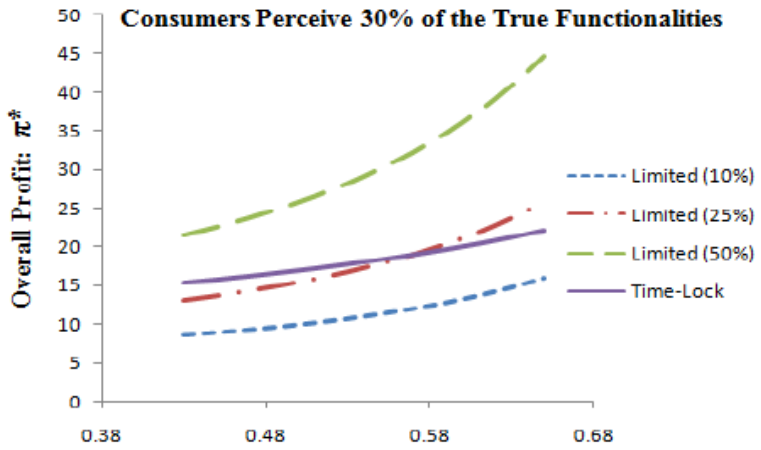


Figure 2-10. Time-lock vs. limited version ($s = 30\%$ of s_0)

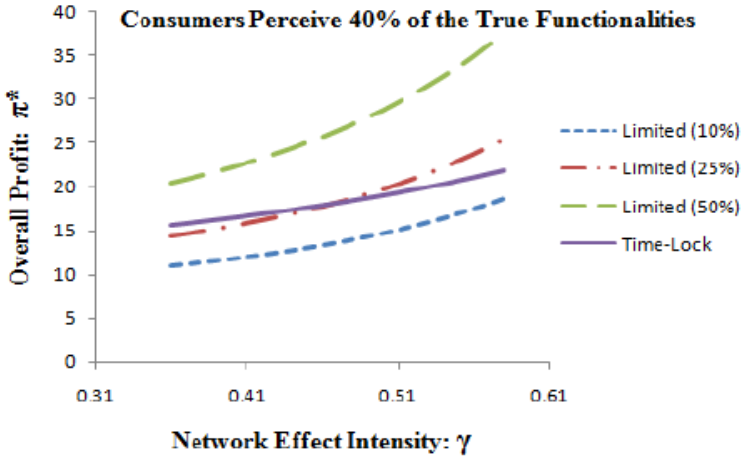


Figure 2-11. Time-lock vs. limited version ($s = 40\%$ of s_0)

Table 2-1. Notation

s:	consumer's prior belief about the quality of the software before free trial
s_t:	true quality of the commercial software
θ:	consumer type, i.e., consumer's preference for the quality of the software
c:	aggregate cost of using the software
t:	the length of time consumers will use the commercial software
τ:	length of the free trial
Q_τ:	total number of buyers that purchase the commercial software given that the length of the free trial time is τ
γ:	the network effect, measuring how much each addition to the number of buyers increases the software's perceived value
P:	price of the commercial software

CHAPTER 3
OPTIMAL SOFTWARE PRICING

Preliminaries

Software industry frequently reports that software piracy causes huge losses of revenue to software firms. According to the Fourth BSA and IDC Global Software Piracy Study (Business Software Alliance 2007), 35% of the software installed in 2006 on personal computers worldwide was obtained illegally, amounting to nearly $40 billion in global losses. To battle the software piracy problem, most countries provide legal protection for software by extending copyright, patent contract and trade secret legislation, and by recognizing software as another type of literary and artistic work subject to intellectual property right (IPR) protection. Given that software piracy is a criminal offense, it would seem that the act of making, distributing, or buying pirated software is simply a question of the lack of morality of the pirate versus the non-pirate (Logston et al. 1994) or the lack of understanding of the copyright laws (Gopal and Sanders 1997). On this basis, the Software and Information Industry Association's (SIIA's) push for increased legislation and enforcement would appear appropriate (SIIA Report 2000). However, there is also evidence to suggest that other factors strongly motivate software piracy. Alongside the lack of censure for piracy and the low likelihood of being caught, the most common reason offered for pirating software is the high cost of legal software (Cheng et al. 1997). This calls for a review of the pricing of legal software that is perhaps a more effective measure within the control of software firms in the fight against software piracy

Previous research on software piracy and copyright enforcement can be broadly grouped into three themes. The first theme has been how the government should respond to information goods piracy by using its various policy instruments. The first instrument of government policy relevant to markets of information goods is a subsidy on purchases of the legitimate item. This

has been advocated as way to discourage copying of databases (Tyson and Sherry 1997) and textbooks (de Freitas 1994), but not commonly observed in the software market. The second instrument of government policy is penalty. Producers of information goods have repeatedly pressed the U.S. Congress to expand the scope of criminal sanctions and raise the penalties for copyright infringement. Responding to industry sentiment, the Congress passed the No Electronic Theft Act 1997 and the Digital Millennium Copyright Act (DMCA) in 1998. Unfortunately, these acts seem to face multiple challenges from the legislation, which largely hinge on the level of scrutiny these acts may apply. The third instrument a government can use to fight piracy is tax, which already has a long tradition in the recorded music industry. Johnson (1985) discusses the implications of a tax on copying, but does not analyze how a producer would adjust its strategy in response to the government policy. Chen and Png (2003) find that tax is social welfare superior to penalty but inferior to a subsidy strategy.

The second theme of software piracy research is how the software producer should respond to piracy through preventive controls. Zwass (1997) points out that "preventive controls (increasing the cost of piracy by technological means) can be used to combat software piracy." The study of how much software producers should engage in selective copyright protection, however, remains ambiguous due to a special characteristic of software, namely, demand-side network externality. From an economic viewpoint, "tolerating some piracy has been shown to have some positive aspects in that piracy makes a product available to those who cannot afford it, increases the consumer base for a product, and creates positive network externalities" (Khouja and Park 2007). Prasad and Mahajan (2003) study the problem of finding an optimal level of protection for a software monopoly. Results from their study indicate that a monopoly should start with minimum protection of its software but impose maximum protection and maintain it

thereafter halfway from the diffusion process. However, due to exogenous legal and social factors, the optimal level of "protection" varies significantly and becomes hard to maintain as the speed of adoption increases. Gopal and Sanders (1997) investigate the effects of preventive and deterrent controls (legal sanctions) on software publishers' profits. Their results suggest that preventive controls do not increase publishers' profits, while deterrent controls can have a positive effect on profitability.

Technological deterrence efforts are often limited in their efficacy (Berst 2002) since they are only effective until the first successful hacker, and legal deterrence relies on enforcement and consumers' awareness of the law (Chellapa and Shivendu 2005). This makes many researchers (e.g., Venkatesh et al 1995, Chen and Png 1999, Peace et al 2003) look into the means of combating software piracy through the most conventional business strategy – pricing (the third stream of the research on software piracy). Specifically, how to price software in the presence of piracy is a critical problem for software firms. A higher than optimal price results in more piracy and revenue loss as "software too expensive" is found as the major reason for consumer piracy (Cheng et al 1997). Further, it leads to an undesired consequence of a smaller installed base and slower diffusion of the software product. A lower than optimal price has the benefit of discouraging piracy at the sacrifice of profit loss. While the industry has been engaged in piracy proofing, academic literature has largely focused on pricing issues without adequately addressing such digital product characteristics as the network effect. In addition, the legitimate demand for digital experience goods, albeit with variable price boundaries, piracy deterrents and quality/feature preferences, has largely been ignored. The purpose of this paper is to fill this gap.

I develop an analytical model that employs the empirical findings on software diffusion by Givon, et al (1995) to examine optimal pricing strategies for a software product under two

coalescing effects of piracy and word-of-mouth through its entire life cycle and evaluate how the change of underlying parameters (e.g., the demand of the innovators, the word-of-mouth effect, the legal conversion rate, etc.) will impact the software firm's pricing decision. Specifically, I incorporate the effect of pricing in my analytical model by explicitly making both the coefficients corresponding to the influence of innovators and the legal conversion rate in Givon, et al (1995)'s model as functions of software price at each time period such that an optimal price can be found in each period to battle against piracy while having software firm's profit maximized. I find that the demand of the innovators has the most significant impact on the firm's pricing decision that takes into account of both piracy and word of mouth effect. My research recommends market skimming pricing strategy if innovators demand is high and market penetration pricing strategy is preferred otherwise. Further, the increase of conversion rate of imitators to buyers makes the penetration pricing strategy more attractive. Most interestingly, the optimal profit from instituting a two prices policy for a software product with five years lifespan outperforms that from a one price policy by no more than 4%, a finding that corroborates the common one price policy observed in reality.

Model Setup

Consider a software firm selling a software product in a market where there are N_t potential customers in the t-th period where $t = 1, \ldots, T$, and T represents the lifespan of the software from its initial launch until the final replacement by major upgrades or other products. Since at most one copy of the software application is needed for each computer, the total number of computers is used as a proxy of the maximum market potential N_t. The price to charge in each period is p_t, $t = 1 \ldots T$. The marginal production cost of the software is assumed to be negligible as is the case for digital products.

It has been established in marketing literatures (e.g., AEA 2002, Givon et al. 1995, Lilen and Rangaswamy 1999) that there are generally two groups of consumers in the software industry – the "innovators" and the "imitators." The "innovators" refer to those early adopters of the product, while the "imitators" purchase the software because of word-of-mouth influence from those who have already adopted the product. The word-of-mouth influence on the diffusion of the software in essence is equivalent to the positive network externality effect in economics (Cabral et al. 1999, Ellison and Fudenberg 1995, Godenberg et al. 2003). Over time, positive word-of-mouth can create a bandwagon effect as the network becomes more valuable and more people join in the positive feedback loop.

Let X_{t-1} and Y_{t-1} be the cumulative number of buyers and pirates at the end of period $t-1$ respectively. Assume that each customer acquires only one software package and there are no repeat purchases. Or, equivalently treat repeat purchases as the demand from customers in different periods. Thus, $N_t - X_{t-1} - Y_{t-1}$ equals the effective market potential for period *t*. Let *a* and *b* represent the "innovation" and "imitation" coefficients, i.e., the proportion of innovators and imitators among consumers, respectively. It is shown in Givon, et al. (1995) that the word-of-mouth effects from both buyers and pirates are the same. Hence, a single word-of-mouth coefficient *b* is sufficient. Since the word-of-mouth effect reflects the adopters' overall experience of the product quality and is insensitive to price, I therefore model the effect of pricing on the diffusion of software through the coefficient of innovation such that:

$$a(p_t) = c - k \cdot p_t \tag{3-1}$$

where *c* corresponds to the proportion of innovators when the price is zero and *k* is the price sensitivity parameter of the innovators. Thus, the total software adoption due to the influence of innovators in the *t*-th period equals to $a(p_t)\left(N_t - X_{t-1} - Y_{t-1}\right)$. The total software adoption due to

word-of-mouth effect in the *t*-th period is described by $\frac{b}{N_t}(X_{t-1}+Y_{t-1})(N_t - X_{t-1} - Y_{t-1})$. This result is derived by multiplying $(N_t - X_{t-1} - Y_{t-1})$, the effective market potential at time *t*, by the influence of word-of-mouth effect, which equals the product of cumulative adopters of both buyers and pirates $(X_{t-1}+Y_{t-1})$ and the coefficient of imitation *b*. The division of N_t amounts to normalizing the expression of word-of-mouth effect to make it scale free.

Assume that out of the software adopters influenced by the word-of-mouth effect, a $\varphi(p_t)$ proportion will buy the software, while the other $1-\varphi(p_t)$ proportion will pirate. $\varphi(p_t)$ corresponds to the legal conversion rate among imitators. Similar to the demand of the innovators, the proportion of buyers is dependent on the price of the software. That is, the proportion of buyers decreases as the price of the software increases. In addition, if the price is zero, there is no need to pirate and all imitators will adopt the software for free. Hence, the following expression captures the proportion of buyers among imitators due to word of mouth effect.

$$\varphi(p_t) = 1 - e \cdot p_t \tag{3-2}$$

Therefore, the number of buyers (x_t) and pirates (y_t) in the *t*-th period can be described in Equation 3-3 and 3-4, respectively. The cumulative number of buyers (X_{t-1}) and pirates (Y_{t-1}) at the end of $t-1$ period are given in Equation 3-5 and 3-6, respectively.

$$x_t = \left(a(p_t) + \varphi(p_t) \cdot b \cdot \frac{(X_{t-1}+Y_{t-1})}{N_t} \right) \cdot (N_t - X_{t-1} - Y_{t-1}) \tag{3-3}$$

$$y_t = (1-\varphi(p_t)) \cdot \frac{b}{N_t} \cdot (X_{t-1}+Y_{t-1})(N_t - X_{t-1} - Y_{t-1}) \tag{3-4}$$

$$X_{t-1} = \sum_{n=1}^{t-1} x_n \tag{3-5}$$

$$Y_{t-1} = \sum_{n=1}^{t-1} y_n \tag{3-6}$$

Figure 3-1 depicts the diffusion dynamics of the software product in the presence of piracy and word of mouth effect in our model.

Now I am in a position to present the general software-pricing model in the presence of both piracy and word-of-mouth effects as follows.

General Model: The Multi-period Multi-price Software Pricing Problem:

The following general model captures the multi-period software pricing problem:

$$\max_{p_t} \sum_{t=1}^{T} \frac{p_t x_t}{(1+\delta)^{t-1}}$$

s.t. (3-1), (3-2), (3-3), (3-4), (3-5), (3-6), and

$$a(p_t) \geq 0, \quad \varphi(p_t) \geq 0 \tag{3-7}$$

$$x_0 = y_0 = 0 \tag{3-8}$$

Software publisher seeks to set the optimal price for each period to maximize the total profit during the software lifespan by taking into account counteracting effects of piracy and word-of-mouth. Note that the revenue in each period is discounted by a factor δ. Given price p_t, the innovation coefficient and the legal conversion rate among imitators in each period are defined in Equation 3-1 and 3-2. In accordance, Equations 3-3 and 3-4 give the total number of buyers and pirates in period *t*. The cumulative software adoption due to legal purchase and piracy are calculated in Equations 3-5 and 3-6. Equation 3-7 prescribes that if the price is set too high, there are no innovators and all imitators will pirate. Finally, Equation 3-8 gives the initial conditions before the launch of the software.

Pricing Strategy in Two Period Model

I begin my analysis with the simplest two-period pricing model and later on extend to more complicated cases. When the planning horizon has only two periods, the general model reduces to the following.

$$\max_{p_1, p_2} \quad x_1 p_1 + x_2 p_2 / (1+\delta) \tag{3-9}$$

$$s.t.$$
$$x_1 = (c - k \cdot p_1) N_1$$
$$x_2 = \left((c - kp_2) + \frac{(1-ep_2)b}{N_2} \cdot (c - kp_1) N_1 \right) \cdot \left(N_2 - (c - kp_1) N_1 \right)$$
$$c - kp_1 \geq 0, \quad c - kp_2 \geq 0, \quad 1 - ep_2 \geq 0$$

In the two-period model, I assume that the market potential N_1 and N_2 are close due to short planning horizon. That is, $N_1 = N_2 = N$. For similar reason, I can ignore the discounting factor in the two-period model. A brief inspection of the model reveals that several observations are in order. First, the market potential N is simply a constant multiplier in the profit function. Consequently, an optimal pricing strategy does not depend on potential market size. Therefore, we can normalize N to 1 without loss of generality. Second, without some initial copies of software available for pirates to pirate, there will be no pirates in the first period, i.e., $y_1 = 0$. Third, the word-of-mouth effect from pirates has no impact on the diffusion of the software in the two-period model since the effect from the pirates in the second period will not be felt until the third period. Bearing these observations in mind, we further simplify the two-period pricing problem as follows.

$$\max_{p_1, p_2} \quad (c - k \cdot p_1) \cdot p_1 + \left[c - k \cdot p_2 + b \cdot (1 - e \cdot p_2) \cdot (c - k \cdot p_1) \right] \cdot (1 - c + k \cdot p_1) \cdot p_2 \tag{3-10}$$

$$s.t.$$
$$c - kp_1 \geq 0, \quad c - kp_2 \geq 0, \quad 1 - ep_2 \geq 0$$

First order conditions require

$$p_1 = \frac{c - bkp_2 + 2bckp_2 + bkep_2^2 - 2bcekp_2^2 + ckp_2 - k^2 p_2^2}{2k(1 - bekp_2^2 + bkp_2)} \tag{3-11}$$

$$p_2 = \frac{c + bc - bkp_1}{2(bce - bekp_1 + k)} \tag{3-12}$$

One can derive optimal prices in both periods by solving Equations 3-11 and 3-12 simultaneously. However, it is analytically impractical to compare the optimal prices in the first and second period due to the complexity of the closed form solutions of p_1 and p_2. To gain more managerial insights, I therefore resort to computational analyses.

Computational Analyses of the Two Period Model

From Equations 3-11 and 3-12, the optimal prices of the software product under a two-period planning horizon can be determined if the value of the four independent parameters – *c*, *k*, *b*, and *e* – are known. Bass (1969) estimates the innovation coefficient $a(p)$ and imitation coefficient b for eleven different consumer durable products. The average for the innovation coefficient is 0.0163 and 0.216 for the imitation coefficient. Lilen and Rangaswamy (1999) study diffusion parameters for 112 products over 10 product categories for different time intervals. For the innovation coefficient, they estimate the average as 0.037 with a median of 0.025, and for the imitation coefficient the average is 0.327 with a median of 0.280. Givon, et al.(1995) study the diffusion of word processor and spreadsheet software in the United Kingdom from January 1987 to August 1992, and find 0.0002 and 0.00069 as the innovation coefficients for word processor and spreadsheet, respectively. They also estimate from empirical data that imitation coefficients b for word processors and spreadsheet software are 0.13518 and 0.10399 respectively. The proportion of buyers due to word-of-mouth effect, the φ in Figure 3-1, is estimated to be 0.14378 for word processor and 0.12122 for spreadsheet software.

Comparing the coefficients from Givon, et al. (1995) and Bass (1969), I find that the innovation coefficient for software products is very small (i.e. $a(p) = 0.0002$), but relatively large for durable consumer products ($a(p) = 0.0163$). Further, the magnitude of word-of-mouth effect ($b / a(p)$) for software products (0.13518/0.0002 = 675.9) far exceeds that of consumer durables (0.216/0.0163 = 13.25). This implies that the word-of-mouth effect plays an important role in the software pricing decision.

Since the diffusion coefficients for word processor and spreadsheet do not differ significantly as they belong to the same category of office productivity applications, in the analyses that follow I will focus on one software product, the spreadsheet, specifically. Hence, I set $b = 0.10399$ in my experiments. The imitator's price sensitivity parameter *e* can also be derived with the knowledge that the spreadsheet software (Excel from Microsoft) was sold at approximately \$230 ($p_{prevail}$) in the United Kingdom from January 1987 to August 1992. Then, from Equation 3-2 I have $e = \frac{1-\varphi}{p_{prevail}} = 0.00382$.

In order to find a close estimation of k, I define $p_{\max}$ as the highest reservation price among innovators in the sense that if the price is set above $p_{\max}$, no innovator will purchase the software. That is, $0 = c - kp_{\max}$. Further, from Equation 3-1, the proportion of innovators, when the price is set at the prevailing price, is given by: $a(p_{prevail}) = c - kp_{prevail}$. Solving these two equations simultaneously, one finds that the parameter *k* is given by $k = \frac{a(p_{prevail})}{p_{\max} - p_{prevail}}$. Simple algebraic inspection indicates that higher reservation price ($p_{\max}$) leads to lower price sensitivity (k) of innovators, which is consistent with common practice. To find a close estimation of $p_{\max}$,

I define p_{high} as the reservation price of imitators in the sense that no imitator influenced by the word-of-mouth effect will purchase the software if the price is set above p_{high}. According to Equation 3-2, $1 - e \cdot p_{high} = 0$. Conceivably, innovators have a higher reservation price than the imitators. Therefore, I specify the maximum allowed price $p_{\max} = m \cdot p_{high}$. For simplicity purpose, I choose $m = 2$ by assuming innovators have a reservation price that is twice as much as that of the imitators. Simple algebraic indicates that $k = \dfrac{a(p_{prevail})}{\dfrac{m}{e} - p_{prevail}} = 0.00000235$.

The only parameter that remains unknown is the zero price demand (*c*) of innovators. Unfortunately, no estimation of *c* is available from the literature. Therefore, a major objective of this section is to investigate the impact of c over the optimal pricing strategy and the total profits for a software firm.

The Impact of c

The parameter *c* in essence characterizes the demand of innovators. Even though I can not find an accurate estimate of *c* in the literature, it is quite clear that the zero price demand should be higher than the demand under the prevailing price. Therefore I let *c* vary between [0.0007, 0.0021] in our experiments, corresponding to one to three times of 0.0069 -- the demand of innovators under prevailing price estimated in Givon et al. (1995).

As shown in Figures 3-2 and 3-3, the firm's optimal profit and optimal first period price increases at approximately a linear rate as *c* increases from 0.0007 to 0.0021. Due to the short planning horizon, there are no pirates in the first period. Therefore, a higher price can be set as the demand of innovators increases. The optimal second period price however suffers from the presence of piracy. When the demand of innovators is weak (i.e $c < 0.0012$), it is profitable for the software firm to lower the second period price to attract imitators. As *c* increases, the optimal

second period price quickly converges to p_{high} and remains constant thereafter, indicating that it is more profitable for the software firm to let all imitators pirate the software and to sell the software to innovators only in the second period. In other words, the benefits of charging a higher price to the innovators offset the benefits of selling to the converted imitators. Apparently setting a second period price higher than p_{high} thus forcing some innovators to pirate, is not optimal to the software firm and hence the optimal second period price remains constant after reaching p_{high}. For the entire range of c, the optimal first period price dominates that of the second period price, suggesting that a market skimming pricing strategy is preferred under this situation. It is known that word-of-mouth effect can become significant only if a certain magnitude of the user base, i.e., a critical mass, has been reached. With only two periods of planning horizon, the installed base of the software is often very small. Therefore the software firm is better off skimming the market, as the word-of-mouth effect is diminutive.

Computational Analyses for the Multiple Period Software Pricing Model

Although software publishers could set an optimal price in each period (e.g., each month), we seldom observe frequent price fluctuations in the software industry. One possibility is that software products, unlike airline tickets, do not have outstanding seasonal demand volatility. Another possible explanation for stable prices is that it is impractical to make frequent changes of price across all distributors if the software is sold through multiple channels. Also, there are psychological effects among consumers that restrict the software publisher from changing the price without a major upgrade of their products, and so on. Therefore, in the computational analyses, I allow only one change of price during the software lifespan. In addition, by comparing the results from the two-price model with those of the fixed price model, I can infer whether the firm should consider more frequent price changes. In sum, the problem is described

as a two-price, multi-period pricing problem. It is my objective to find optimal prices in each stage (p_1 and p_2) and the best timing to change the price. More importantly, I would like to answer the fundamental question: should the software publisher adopt a market penetration pricing or a market skimming pricing strategy under a long planning horizon?

Past literature estimates the average lifespan of a software package to be between 3 to 7 years. For example, Strassmann (1998) indicates that for a typical corporation, average software maintenance costs range from twenty to thirty percent of annual spending on software purchases and concludes that average software lifespan would be around five years. Givon, et al. (1995) lists the monthly data of the number of total PCs and excel users in 68 periods. Thus, I analyze the diffusion of software adoption up to 60 periods (months), i.e., 5 years.

Pricing Strategy for the Two Price Multi Period Case

In this section, I explore the impact of the proportion of buyers due to word-of-mouth effect (φ) and the demand of the innovators (c) on the optimal price and profit for a software firm. An exhaustive search over the solution space is conducted to find optimal prices as well as the optimal timing for changing the price. In addition, we compare the result against the benchmark case where the price remains unchanged during the entire software lifespan to analyze the impact of price change on the firm's profitability. To study short-term versus long-term effects, we run analyses with various software lifespans.

Through computational analyses, I seek to address the following questions: (1) Should software firm adopt a market skimming pricing or market penetration pricing strategy; and (2) how much is the profitability improved by choosing the two-price model over the fixed price model. The remaining discussions will still focus on spreadsheet software specifically.

The impact of *c*

Computational analyses show intriguing and yet intuitive results of how innovator's demand affects the pricing strategy of the software firm when the software lifespan increases from 2 periods to 60 periods with one period corresponding to one month. As shown in Figure 3-4, the optimal prices in both stages increase as c increases. Penetration pricing strategy is preferred when the demand of innovators is weak (i.e. $c < 0.0012$) and skimming pricing strategy is favored when the demand of innovators is strong (i.e. $c > 0.0012$). When the demand of innovators falls below a critical value (i.e. $c < 0.0012$), it puts a potential threat to the diffusion of the software. Under such a situation, it is more profitable for the software firm to have a lower first stage price (often for a short time) to quickly build up the user base and raise the price in the second stage to take advantage of the word of mouth (i.e., positive network externalities) effect. However, when the demand of innovators is relatively strong, it is more profitable for the software firm to skim the innovators by setting a higher price in the first stage. At a later stage, however, when the software becomes readily available and it is harder to control piracy, the firm lowers its price with an aim to sell more copies at a cheaper price to the imitators. Finally, when the demand of innovators is stronger as reflected by an increasing value of c, the optimal total profit grows accordingly as shown in Figure 3-5.

The impact of φ

The proportion of buyers due to word-of-mouth effect represents the percentage of those imitators influenced by the word-of-mouth effect who "convert" to buyers. Consequently, $(1 - \varphi)$ proportion of those imitators choose to pirate the software. From extensive computational analyses, we observed that the change of φ never significantly alters the pricing strategy within the feasible range of parameter values. Instead, an optimal pricing strategy is primarily pre-

determined according to the chosen value of c. Figures 3-6 and 3-7 plot the optimal first and second stage prices with various φ's under either strong demand (c = 0.0014) or weak demand (c = 0.001) of innovators. We choose φ in the range of $[0.08, 0.16]$ for the shown experiments as it is consistent with the empirical estimates of the legal conversion rate for spreadsheet software. As φ increases, the firm realizes more profit since more imitators prefer to purchase rather than pirate. At the same time, I observe that the optimal prices in each stage are also increasing in φ. This result is intuitive, as a higher "conversion" rate of imitators implies less price sensitivity, which leads to a higher price set by the firm. Once again, I find the skimming pricing ($P_1 > P_2$) is favored when the demand of innovators is strong (i.e. c = 0.0014) but the gap between the optimal first and second stage prices is shrinking as φ increases, suggesting the penetration strategy is making up the ground and possibly become optimal for large enough φ. Similarly, we find that penetration pricing ($P_1 < P_2$) is preferred when the demand of innovators is weak and the gap between P_1 and P_2 enlarges indicating the penetration strategy will remain optimal as φ increases. Under both situations, the increase of the legal conversion rate (φ) make penetration strategy more attractive to the software firm.

The impact of software lifespan

Figures 3-8 and 3-9 depict optimal prices and profits for various software lifespans. When the planning horizon increases, the software firm charges a lower price but bringing in a higher profit. A possible explanation of this result is that with a longer software lifespan, the network externality effect becomes significant such that it pays off to set a lower price to build up a larger user base and sell to more people who are influenced by the word-of-mouth effect. Once again, I find that optimal pricing strategy is not affected by the change of the software lifespan (for those software with more than 24 months planning horizon).

Pricing strategy: Market penetration or market skimming?

Using the coefficient values derived from past empirical studies, I find from all computational analyses that the market skimming pricing is always preferred for software products when the demand of innovators is high. Penetration pricing is favored otherwise. Figures 3-4, 3-6, 3-7 and 3-8 provide examples of prices with respect to different values of c, φ, and software lifespan. This result is rather intuitive as one would expect the firm to set a lower price when introducing a software product in order to benefit from positive network externalities under a low demand, but set a higher price in the first stage to make more profits and deliberately restrain the diffusion of the software to avoid piracy under a high demand.

Optimal Timing of Price Change

Figure 3-10 shows an example of optimal timing of price change with different software lifespans. We observe that when the planning horizon increases, the firm should hold the optimal first stage price for a longer time before switching to a different price in the second stage. In addition, the optimal timing of price change often lies between one quarter and one third of the total lifespan of the software.

Two Price vs. One Price Benchmark

In the one-price benchmark case, the software firm finds the single best price that will result in the highest possible profit throughout the software lifespan. I observe from extensive computational experiments that the optimal single price is in between the optimal prices in the two-price model. Figure 3-11 shows the optimal single price and the optimal two prices for the same parameter setup.

It is quite clear that the software firm will realize less profit from implementing one-price than from the two-price policy. Figure 3-12 plots the percentage of profit gain by adopting the two-price policy versus the one-price benchmark policy with the same parameter setup. Given

the baseline parameters conforming to empirical data, a somewhat striking result from Figure 3-12 is that the profit difference between two-price policy and one-price benchmark case is rather insignificant as evidenced by the profit gain never exceeds 4%. This finding lends support for the one fixed pricing of software packages commonly adopted in the industry. Also note that, the curly shaped profit gain pattern is consistent with the switch of the pricing strategies from penetration to skimming as c increases. The first decreasing then increasing profit gain curve corresponding to the dying out of the penetration strategy and then the rise of the skimming strategy.

Summary

In Chapter 3, I examined the pricing policy of a software firm in a market with both piracy and word-of-mouth (positive network externalities) effects. While prior literature has studied the piracy problem from various perspectives, the implications of both the piracy and network effect on the diffusion of the software and hence the profitability of the software firm in a multi-period setting has not been analyzed. Incorporating the empirical findings of Givon et al. (1995), I show how a software firm can develop an optimal pricing policy that maximizes the total profit for its software products with various lifespans.

As commonly modeled in the past literature about new product diffusion, I divide consumers into two groups (innovators and imitators) based on their affinity toward piracy. My extended diffusion model shows that potential consumers at each time period t are influenced by the size of the current installed base (including legal and pirate users), the price and the word-of-mouth effect simultaneously. Ideally, the software firm foresees the impact and varies the price of its product in each period to battle against piracy while having its profit maximized. In reality, however, we do not observe frequent price fluctuations in the software industry. As a result I turn my analyses to the settings where only one price change is permitted and focus on the impact of

the change of parameters (e.g., the word-of-mouth, software lifespan etc.) over the firm's decision whether to adopt market penetration or skimming pricing strategy. Our research recommends that market skimming strategy is desirable if the demand of innovators is high and penetration strategy is preferred otherwise. We also find that the change of other parameters such as the word-of-mouth effect, the lifespan of software never significantly alters the pricing strategy within the feasible range of parameter values, suggesting that the demand of innovators, especially the zero price demand of innovators plays a more influential role.

My extensive computational analyses provide practical value to the industry practitioners as the setup of the experiments are based on empirical findings of extant research. Like all information goods, software products are easy and cheap to copy and distribute. This research addresses the fundamental question of weighing the gain from positive network externality and the financial loss due to piracy and provides further guidelines on how the firm should price their products in the long run.

I find that there is a lack of empirical research on the zero price demand of innovators and how price sensitive they are to software, which should be worthwhile topics for future research. The recent move toward cloud computing and software as a service will certainly present new challenging research opportunity in the area of software pricing and piracy.

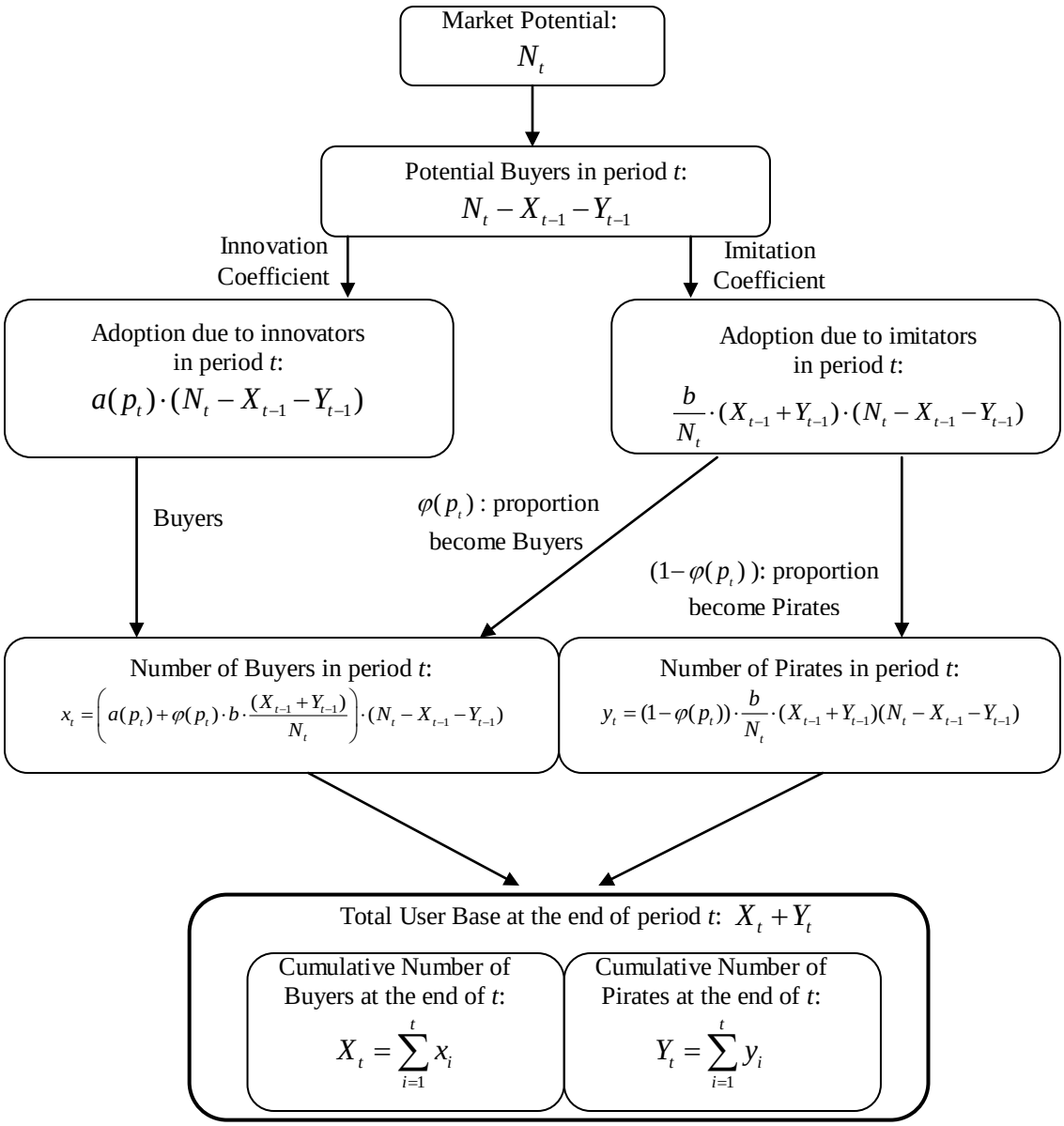


Figure 3-1. Legal and illegal software diffusion over time

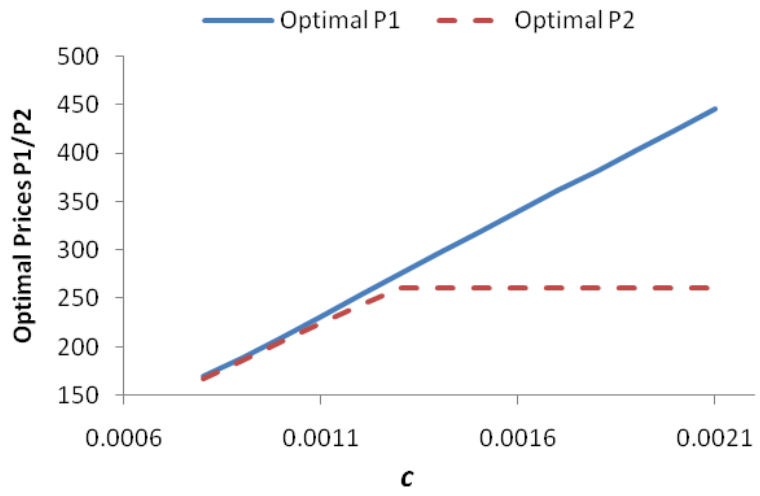


Figure 3-2. Optimal prices under various *c* (2 periods)

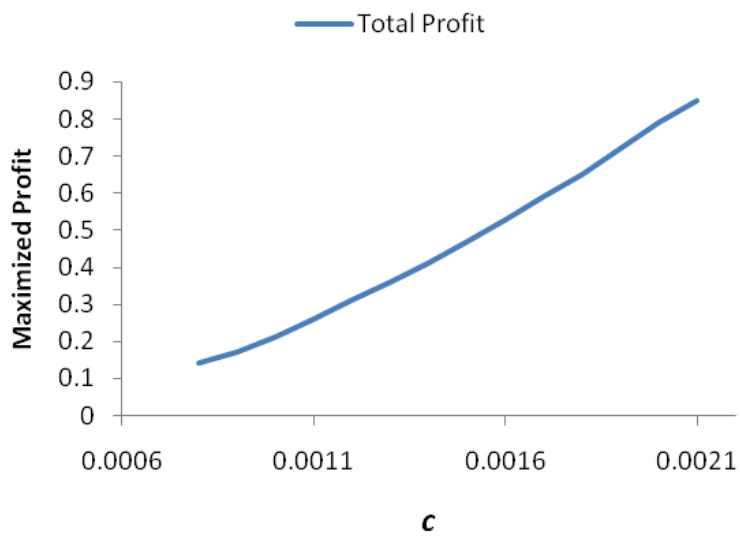


Figure 3-3. Optimal profits under various *c* (2 periods)

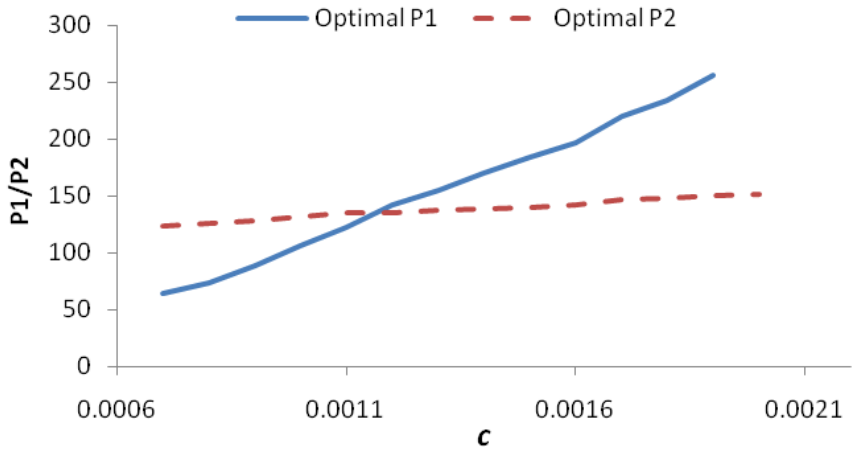


Figure 3-4. Optimal prices under various *c* (60 periods)

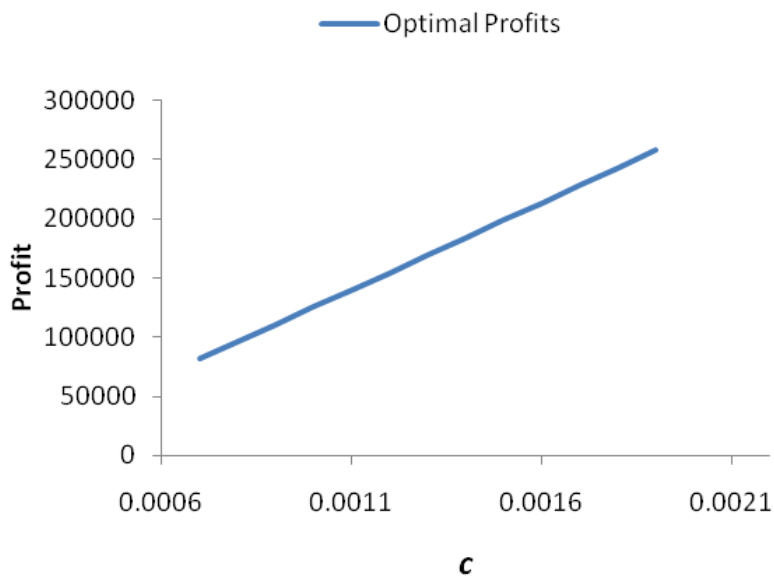


Figure 3-5. Optimal profits under various *c* (60 periods)

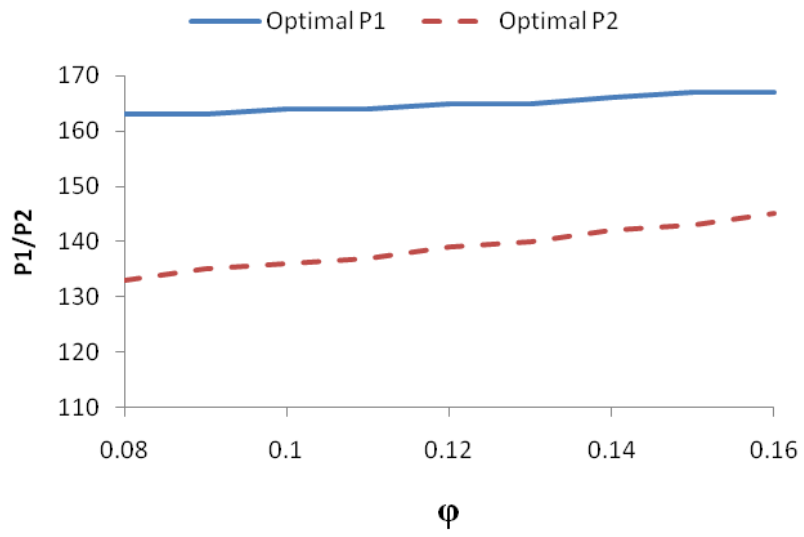


Figure 3-6. Optimal prices under various φ with strong demand (60 periods, c=0.0014)

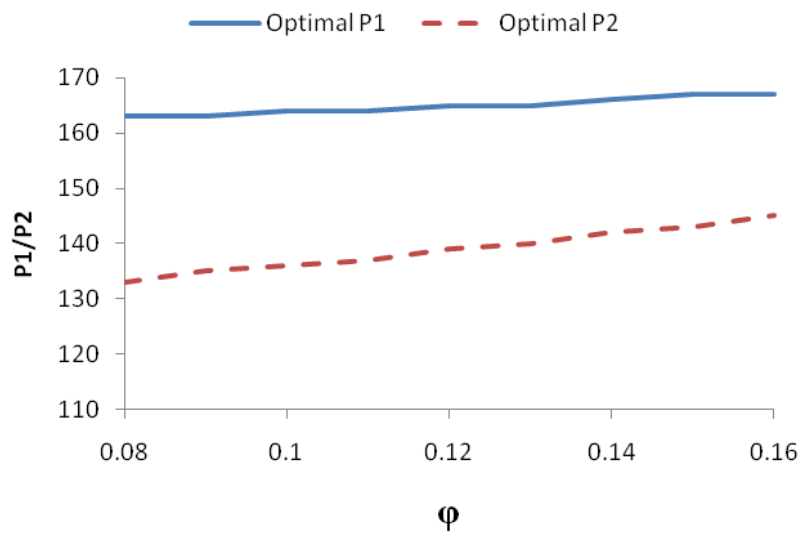


Figure 3-7. Optimal prices under various φ with weak demand (60 periods, c=0.0010)

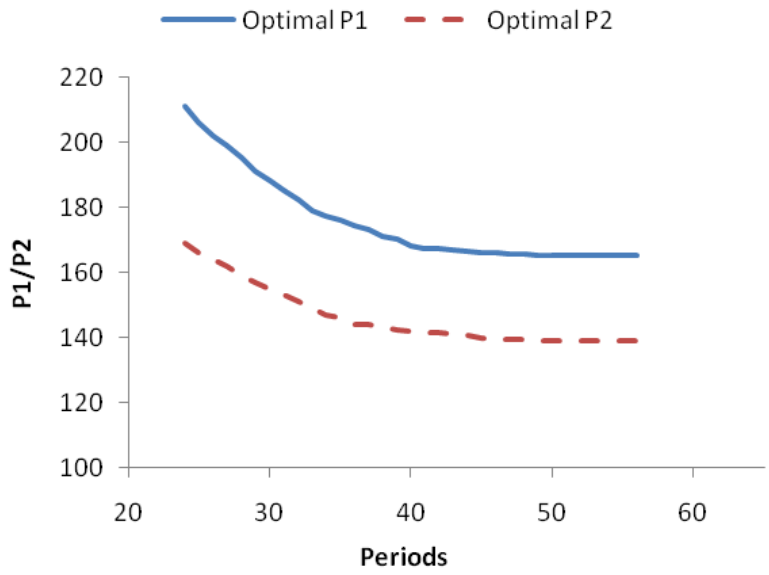


Figure 3-8. Optimal prices under various lifespan

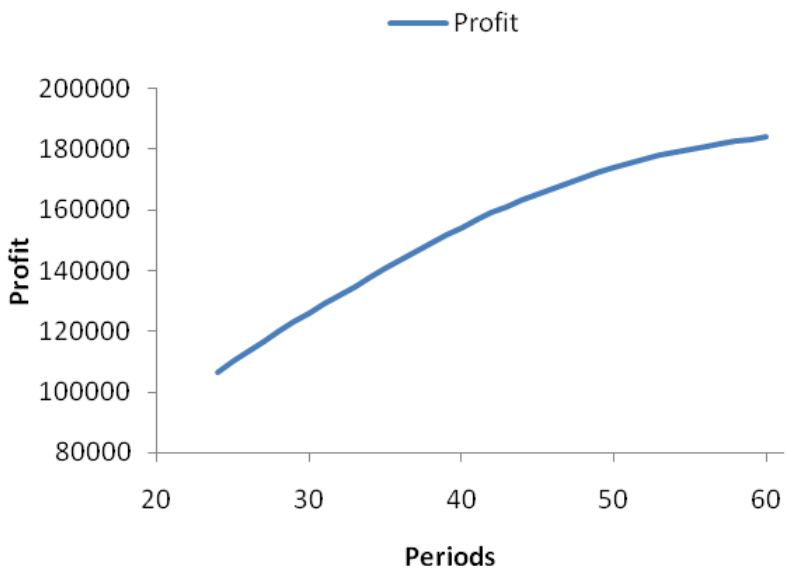


Figure 3-9. Optimal profits under various lifespan

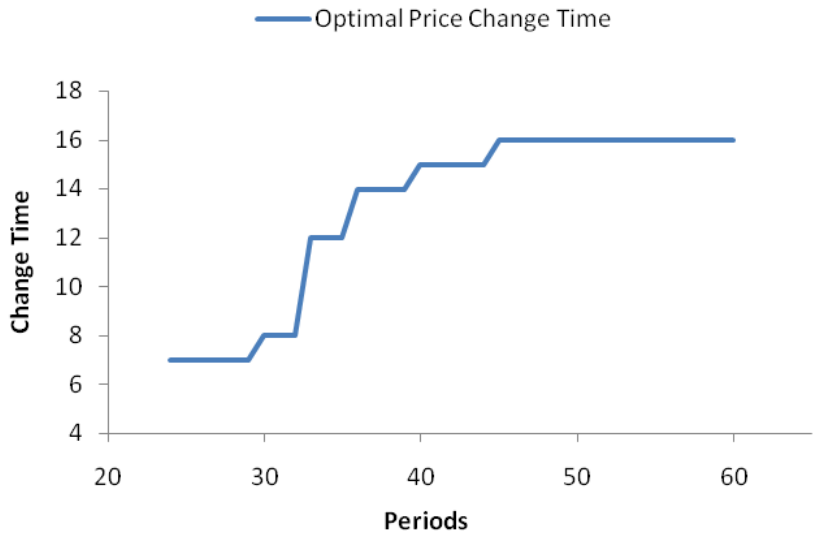


Figure 3-10. Optimal timing for price change

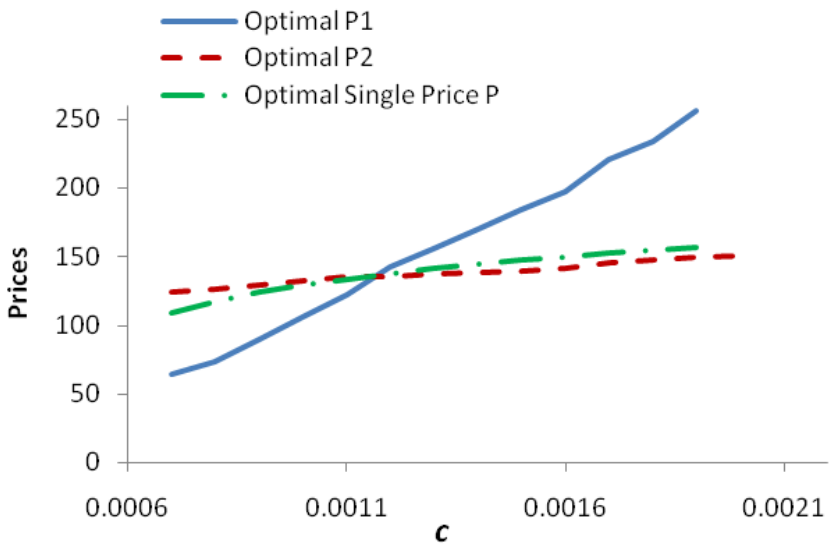


Figure 3-11. Optimal single price(p) vs. two prices strategy (p1, p2) under various *c*

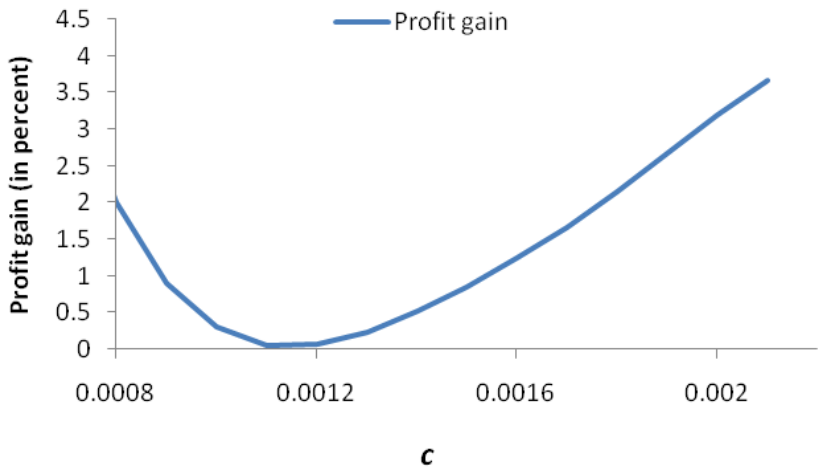


Figure 3-12. Profit gain from two-price policy under various *c*

CHAPTER 4
THE IMPACT OF NETWORK EXTERNALITIES ON THE COMPETITION BETWEEN OPEN SOURCE AND PROPRIETARY SOFTWARE

Preliminaries

Open source software (OSS) is one of the most important recent developments in the software industry. OSS refer to those programs "whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or the modified program without having to pay royalties to previous developers" (Wheeler, 2003). OSS has been forecasted by Price Waterhouse Coopers as one of the five significant technology trends in the software sector during the next decade, which include peer-to-peer architecture, the rise of OSS, intelligent software agents without constant human supervisor, software development based on the Model Driven Architecture, and a reversal of the trend toward the use of Web browser as the standard client for client/server applications (Datta, 2003).

The OSS movement was originated by Richard M. Stallman of the Artificial Intelligence Lab at MIT in the 1970s. With an intension to build a kernel better than the existing UNIX system, Stallman started developing free tools and software applications and later initiated the open source movement. Open source software is generally licensed under the GNU General Public License (GPL), "which allows free use, modification, and distribution of the software and any changes to it, restricted only by the stipulation that those who received the software pass it with identical freedoms to obtain source, modify it, and redistribute it" (Rosenberg, 2000). The definition of open source software (OSS) not only allows free access to the source code but covers the following aspects such as free redistribution, derived works, integrity of the author's source code, no discrimination against persons or groups, no discrimination against fields of endeavor, distribution of license, license not to be specific to a product, license not to restrict

other software, and the license not to be predicated on any individual technology or style of interface, see http://www.opensource.org for further details.

Increasing number of OSS products have become available in the market, ranging from common office suites such as OpenOffice.org™ (formerly known as StarOffice) by Sun Microsystems, database systems such as mySQL, to thousands of specialized scientific applications. The most notable OSS products are without a doubt Linux operating system and Apache web server software. Although the presence of Linux on the average desktop is not yet significant, Linux grows to be a major threat to its commercial counterpart – Microsoft's Windows Server – in the enterprise server market. Projected to have 30 percent of the server market by 2008, Linux accounted for 25.7 percent share of the server operating system market in 2007, up from 15.6% in 2003 according to a research by IDC. The growth rate of Linux's acceptance has been phenomenal as well. IDC recently reported that Linux-based server sales is expected to reach US$9.1 billion in 2008, up 22.8 percent from the previous year compared to 3.8 percent growth rate for the overall server market. On the Web server front, the open source Apache has always been the dominant number one choice since Apache grew into the leading Web server in April 1996, with over twice the market share of its commercial competitor Microsoft IIE (according to www.netcraft.com)

The OSS gains escalating popularity among corporations because it is cheaper to adopt, less vulnerable to viruses and other rogue programs, more reliable and scalable, and most importantly OSS enables users to adapt the software to their needs with the access to the source code. Several factors contribute to the lower total ownership cost of using OSS. For instance, "OSS costs less to initially acquire, upgrade and maintain, and runs more efficiently on older hardware among others" (Wheeler, 2003). Cybersource found that using OSS saves 24 to 34

percent of the total ownership cost over a three-year time span compared with using Microsoft's proprietary approach for a typical organization of 250 computer-using employees (www.cybersource.com).

Although the rise of OSS to prominence signifies the start of fundamental changes to the software industry, extant literature on various intriguing issues of OSS has been sparse. Pioneering research on OSS to date focuses mainly on why developers participate in OSS projects (Lerner and Tirole 2001 and 2002, Hann et.al 2002 and 2004), the incentives of firms to adopt open source initiatives (Mustonen 2003) and the success factors of OSS projects (Fitzgerald and Kenny 2003; Crowston, Annabi, and Howison 2003; Chengalur-Smith and Sidorova 2003; Sagers 2004).

As OSS proves to be a viable technology solution for businesses, another stream of OSS research turns attention to the competition dynamics between OSS and proprietary software (PS). This stream of research includes analysis – both empirical and theoretical – of the public and free nature of OSS products and their impact on the marketplace for software products. For example, Casadesus-Masanell and Ghemawat (2006) modeled the competition between Windows and Linux as a dynamic "mixed duopoly", where a not-for-profit competitor (Linux) interacts with a for-profit competitor (Windows). They showed that, as long as Windows' pricing decision was not myopic, the result of competition would be either the coexistence of the two products or Linux being driven out of market. Chen et al. (2005) employ an agent-based simulation approach to study how OSS can effectively compete with its commercial counterpart. Chen, Cheng, Hsu, and Koehler (2005) empirically test the diffusion of an OSS Web server and Microsoft IIS Web server in the competitive marketplace.

This research aims at a similar topic but differs from the past literatures in two aspects. Firstly, my work focuses on the impact of network externalities on the competition rather than the result of competition between open source and proprietary software. I model the heterogeneous consumers' preferences for products depend on two major factors: the software quality and the network size and I allow the extent of network effect to be different between OSS and proprietary software. Secondly, I investigate the strategic choices and incentives for compatibility between OSS and proprietary software venders – a topic majorly overlooked in previous studies on open source software.

Software products including the OSS have several distinct characteristics when compare to traditional physical goods. Most notably, software exhibits positive network externalities. A positive network effect, or network externalities, refers to the increase of consumer's utility when more consumers become the users of the same product. The positive network externalities effect of software may lead to a software product (not necessarily the higher quality one) to enjoy a commanding market share. A well-known example in the personal computer industry is perhaps the competition between Apple's Macintosh OS and Windows operating systems of personal computers before Windows became the de-facto standard. While Macintosh was touted as a better operating system than competing Windows operating systems, a potential buyer of personal computer tend to adopt the Wintel platform, i.e., Windows operating system with Intel CPU, as there were far more users of Wintel than Apple's personal computers. The more users Wintel attract, the more software developers write applications for this platform, which in turn increases the value of the Wintel platform and makes it more attractive to potential consumers. This positive network externalities phenomenon can be more easily understood as "positive

reinforcement," a term coined by the founder of Microsoft, Bill Gates, in his best seller "The Road Ahead".

Since OSS does not provide formal technical support and relies on the backing of a volunteer community, the network externalities will play a critical role for OSS to effectively compete with its commercial counterpart. Compare to proprietary software, OSS has several distinct advantages due to the positive network effect. For example, the widespread peer review process involved in open source development makes OSS more error-free and resource-efficient than proprietary software. In addition, the right to redistribute modifications and improvements to the code and to reuse other open source code permits all the advantages due to the modifiability of the software to be shared by large communities. These advantages provided by OSS to each individual customer obviously come from the contribution of every other user in the network. Hence, compared to proprietary software users, OSS users might derive more utilities from one more added consumer who becomes the user of the same OSS product. In my model, I therefore allow the possibility of OSS exhibiting a different (possibly stronger) network externalities effect than the proprietary software product, whereas almost all past literature (Lee and Meng 2005, Sen 2007, Asundi et al. 2008) assumes the two share the same network effect.

To survive and win the battle of market shares, OSS and proprietary software vendors may strategically choose to have their products to be compatible or incompatible with its rival software. For example, Windows, a proprietary software product, is incompatible with Linux, an open source software product. In the case of web browsers however, Internet Explorer is completely compatible with its open source rival Mozilla Firefox in the sense that files created for IE users can be used by Firefox users and vice versa. Different compatibility strategies will

lead to different network sizes, and thus result in different network generated values and profits for both OSS and proprietary software venders.

The difference of network externalities effect intensities associated with different compatibility strategy between OSS and proprietary software present a series of research questions: Does the positive network effect always have a positive impact on both OSS and proprietary software? If not, which party will benefit and which party will be hurt? How would the choice of compatibility affect network externality's impact? Furthermore, which party has the most incentive to make its product compatible with its rival in the presence of network externality? In Chapter 4, I address these research questions by separating my analysis into four different cases (see Table 4-1) depending on whether the rivalry software are compatible with each other and whether they share the same network effect intensity.

The main findings of this paper are as follows. When the market is fully covered, i.e., all consumers purchase one of the two products (OSS or PS), network externality has positive impact (i.e. increase the market share) on proprietary software but negative impact on OSS software under Cases 1, 3 and 4. Depending on the underlying parameters, there are mixed results regarding the impact of network externalities under Case 2. Specifically, a threshold value can be found such that if the ratio between the two network effects is greater than the threshold value, then OSS benefits from the network externality; otherwise, proprietary software benefits from the network effect. Furthermore, contrary to the results in Lee and Meng (2005), I find that OSS vender has the most incentive to make its product compatible with its opponent rather than the other way around.

Due to a non-negligible learning cost incurred by getting familiar with the software product, software users may derive negative utilities and hence choose not to adopt either OSS or

PS. Such an action will lead to a partially covered market for my analysis. In addition, most OSS venders often offer another type of software together with their freely available OSS, where they sell a non-divisible bundle of a higher quality OSS (denoted by OC) with commercial support services. In the later section of Chapter 4, I then examine the impact of network externality in a partially covered market by adding OC into competitive landscape and taking learning cost into consideration. Analytical and computational results show that network externality has positive impact on OSS if it is made compatible with proprietary software but negative impact if it is not. OSS software vender is found once again to have the most incentive to make its product compatible with proprietary software.

The Model

Assume that the total number of potential customers who will adopt either open source software (OSS) or proprietary software is *N*, which is normalized to 1 without loss of generality. Let θ represent a customer's preference for software quality and θ is uniformly distributed between $[0,1]$. In the absence of network externalities, a customer derives $u = \theta \cdot s - p$ utility of adopting a software of quality *s* charging price *p*.

Let γ captures the intensity of the network externalities effect. When *Q* is the installed base of the software (i.e., the total number of users adopting the software), network externalities effect increases each consumer's preference (θ) by γQ in accordance with the increase in willingness to pay when an additional user joins the network. When purchasing a software product of quality s in a network of size Q at price p, consumer θ obtains net utility $u = (\theta + \gamma Q)s - p$ under the influence of network externalities. In Farrell and Saloner (1986), $\theta \cdot s$ is the product's "*network-independent*" or standalone value, which depends on both consumer's preference and product quality. The "*network-generated*" value is γQs and exhibits

complementary effects between network size and quality. Therefore in the same network, a high quality product has both higher standalone and network values than a lower quality one. In Katz and Shapiro (1985) and most other research on network externalities, product are homogeneous and consumers differ in their total willingness to pay. The network-generated benefit therefore only depends on the network size.

Let s_o and s_p describe respectively the quality of open source software and proprietary software. Generally, software quality can be measured in several dimensions, such as performance, ease of use, customer support, reliability and security, etc. For the sake of generality, my study is not restricted to any particular dimension. Rather, I define software quality as the characteristics of software other than the price. For practical purpose, I assume $s_p > s_o$, since *all* customers will adopt OSS if the quality of the free OSS exceeds that of the commercial counterpart. In reality, we indeed observe certain quality advantages enjoyed by the proprietary software, such as more user-friendly interface, more reliable and professional customer support, continuous product upgrade and so forth. Let Q_o and Q_p denote the installed user base of open source software and proprietary software respectively. Table 4-2 summarizes the notation used in this paper.

Competition between OSS and Proprietary Software (Full Market Coverage)

I first present the case where there is no network externalities as a benchmark, followed by analyses of competition between OSS and proprietary software in the presence of network externalities under different scenarios. Several key propositions are derived to examine the impact of network externalities on the competition.

Benchmark Case: No Network Externalities

In the benchmark model (see Figure 4-1), I assume that the market is fully covered, i.e., all consumers choose to use one of the two software products. This is always true when the benefit of the product is sufficiently large. In addition, I assume that the open source software is freely availabe hence there is no price component in the net utility. Without network effect, a consumer of type θ derives net utility $u_o = \theta \cdot s_o$ by using the open source software and net utility $u_p = \theta \cdot s_p - p$ by purchasing the proprietary software. Suppose consumer at $\theta_b \in [0,1]$ is indifferent between the open source and proprietary software products, then one has: $\theta_b s_p - p = \theta_b s_o$.

Thus, $\theta_b = \dfrac{p}{s_p - s_o}$, and the resulting installed bases for OSS and proprietary software are θ_b and $1-\theta_b$ respectively. In contrast with conventional Hotelling model, only one party – the proprietary software producer aims at maximizing profit and the open source software vender is passive. Due to the neglegible marginal cost of production, proprietary software firm solves the following profit maximization problem:

$$\max_p \; p(1-\theta_b)$$
$$s.t. \quad 0 < p < s_p - s_o$$

Accordingly, the optimal price of the proprietary software in the benchmark case is, $p_b^* = \dfrac{s_p - s_o}{2}$, and the demands of proprietary and open source software and the profit of the proprietary software vender are given by $Q_{bp}^* = Q_{bo}^* = \dfrac{1}{2}$, $\pi_b^* = \dfrac{s_p - s_o}{4}$, where the subscript *b* denotes the benchmark case of no network externalities, *p* for proprietary software, and *o* for

OSS. Even though proprietary software is of higher quality, since OSS is free, the benchmark-level demands (e.g. installed user base) of proprietary and open source software are the same.

The Impact of Network Externalities on A Fully Covered Market

I first present how the consumers' utility functions of open source and proprietary software change according to different network effect intensities and compatibility strategies. Next, I summarize the impact of network externalities by comparing the equilibrium demands of OSS and proproiteray with the benchmark demands reported in the previous section. By comparing the equilibrium demands among cases before and after the competing software products are made compatible with each other (e.g. Case1 vs. Case 3 and Case 2 vs. Case 4), I am able to find which party (open source or propritery software vender) has the most incentive to make its product compatible with its competitor's product.

Case 1: Both OSS and proprietary software are compatible with each other and have the same network externalities effect

When the rivalry products are compatible with each other, the network size for either software product, where the network value is generated, is equivalent to the sum of the two networks (i.e. $Q = Q_o + Q_p = 1$, since the market is fully covered). In addition, both OSS and proprietary software exhibit the same network effect intensitiy in Case 1 (i.e., $\gamma_o = \gamma_p = \gamma$). Therefore, in the presence of direct network effect, the net utility for OSS and proprietary software consumers are given by:

$u_o = (\theta + \gamma Q) s_o$ and $u_p = (\theta + \gamma Q) s_p - p$, where $Q = 1$.

Marginal consumer θ_o who is indifferent between open and proprietary software is characterized by:

$$(\theta_o + \gamma Q) s_o = (\theta_o + \gamma Q) s_p - p \tag{4-1}$$

The demands of the proprietary and open source software are therefore defined as:

$$Q_o = \theta_o \text{ and } Q_p = 1 - \theta_o \tag{4-2}$$

From Equations 4-1 and 4-2, the marginal consumer θ_o can be derived as:

$$\theta_o = \frac{\left(s_o - s_p\right)\gamma + p}{s_p - s_o} \tag{4-3}$$

Accordingly, the number of open source users and proprietary software buyers are

$$Q_o = \frac{\left(s_o - s_p\right)\gamma + p}{s_p - s_o}, \text{ and } Q_p = \frac{\left(s_p - s_o\right)\left(1+\gamma\right) - p}{s_p - s_o} \tag{4-4}$$

The comparative statics of Q_o and Q_p with respect to p reveals that

$$\frac{\partial Q_o}{\partial p} = \frac{1}{s_p - s_o} > 0 \text{ and } \frac{\partial Q_p}{\partial p} = \frac{-1}{s_p - s_o} < 0 \tag{4-5}$$

Inequality 4-5 shows that if the price of proprietary software increases, the demand for the proprietary software decreases, while the demand for the open source software increases. In order to guarantee non-negative demand for OSS, the following condition must hold:

$$\gamma \le \frac{p}{s_p - s_o} \tag{4-6}$$

The condition in Inequality 4-6 implies that the impact of positive network externalities on consumer's net utility is bounded above by $\frac{p}{s_p - s_o}$. Similarly, in order to guarantee non-negative demand for proprietary software, feasible range of proprietary software price must be within

$$\gamma\left(s_p - s_o\right) \le p \le \left(1+\gamma\right)\left(s_p - s_o\right) \tag{4-7}$$

Proprietary software vender maximizes its revenue by setting the price optimally. The decision problem of the proprietary software vender is therefore formulated as:

$$\max_p pQ_p = p \cdot \frac{(s_p - s_o)(1+\gamma) - p}{s_p - s_o} \tag{4-8}$$
$$\text{s.t.} \quad \gamma(s_p - s_o) \le p \le (1+\gamma)(s_p - s_o)$$

Kuhn-Tucker conditions lead to

$$p^* = \frac{(s_p - s_o)(1+\gamma)}{2} \tag{4-9}$$

I then plug in the optimal price into Equations 4-4 and 4-7 to calculate the optimal demands (i.e., installed base) of the proprietary and open source software and the valid range of γ as follows:

$$Q_o^* = \frac{1-\gamma}{2},\ Q_p^* = \frac{1+\gamma}{2} \tag{4-10}$$
$$0 \le \gamma \le 1$$

The profit of the proprietary software firm is accordingly:

$$\pi_p^* = \frac{(s_p - s_o)(1+\gamma)^2}{4} \tag{4-11}$$

By comparing the above results to the benchmark level demands and profit, I have the following proposition. Proofs of all propositions that follow are delegated to the Appendix.

Proposition 4-1 *In case 1, the installed base of proprietary software increases, the installed base of open source software decreases, both the optimal price and the optimal profit of proprietary software raises in the presence of network externalities.*

When the network effect intensity is different

Unlike past literature, my model allows the possibility that the OSS has higher network effect intensity than the proprietary software. To address this issue (i.e. Cases 2 and 4 in Table 4-1), I use different parameters γ_o and γ_p ($\gamma_o > \gamma_p$) to capture the different network effect intensities for OSS and proprietary software respectively.

When software products are not compatible

Users in OSS network will no longer be able to benefit from the contribution of the users in the proprietary software network and vice versa. As a result, the network size, from which the network value is generated, shrinks for both software products. To address this issue (Cases 3 and 4 in Table 4-1), I use Q_o and Q_p to denote the installed base of OSS and proprietary software respectively. Full market coverage leads to $Q_o + Q_p = 1$.

Following the same approach used in Case 1, where proprietary software vender aims at maximizing its profit and OSS vender reacts passively, Table 4-3 reports the optimal outcomes of the open source and proprietary software under different scenarios. A series of key propositions are thus derived from Table 4-3.

The effect of network externalities: Positive or negative?

By comparing the optimal outcomes of OSS and PS under case 3 and 4 with the benchmark results, I find network externalities have positive impact on the demand and profit of proprietary software, but negative impact on the demand of open source software when software products are incompatible with each other.

Proposition 4-2. *When OSS is incompatible with its proprietary software counterpart, the installed base of open source software decrease, the installed base of proprietary software increases, the optimal price of proprietary software drops, but the optimal profit of proprietary software improves in the presence of network externalities.*

Comparing the optimal outcomes of Case 2 with benchmark results, however, shows mixed results. I find that if the ratio of network effect intensities between OSS and proprietary software (i.e. $\frac{\gamma_o}{\gamma_p}$) falls short of the inverse quality ratio between the two (i.e. $\frac{s_p}{s_o}$), then

proprietary software benefits from the presence of network externalities; otherwise, OSS is favored by more users in the presence of network externalities. These findings can be summarized by the proposition below.

Proposition 4-3. *Define the threshold value* $T_1 = \frac{s_p}{s_o}$. *In case 2, if* $\frac{\gamma_o}{\gamma_p} < T_1$, *the installed base, the optimal price and the overall profit for proprietary software all increase and the installed base of OSS decreases in the presence of network externalities. The results, however, are reversed if* $\frac{\gamma_o}{\gamma_p} > T_1$.

What is the best strategy: Compatible or incompatible?

For either OSS or proprietary software vender, making its product compatible to its competitor allows the user of that product benefit from the network of its opponent, which therefore adds to the value of the product and increases consumer's willingness to pay and their net utility of adopting the software. However, in a fully covered market, the increase of one party's user leads to the decrease of the other party's user and the best response for the competitor in such a situation is to make its product compatible too. Therefore in equilibrium we will see software products are either incompatible with each other or fully compatible with each other. One-way compatibility (one product compatible with the other, but not the other way around) is therefore ignored in my analysis. Furthermore, I am more interested in the question about which party has the most incentive to make its product compatible. The answer to this question can be found by a head to head comparison (e.g. Case 3 vs. Case 1, Case 4 vs. Case 2) between the payoffs (e.g. market shares and profits) for both OSS and proprietary software venders when the software is made from incompatible to compatible with its rival product.

Whichever party receives the highest boost in its payoffs is deemed to have the most incentive to make the change.

Proposition 4-4. *Whether OSS and proprietary software exhibit the same network effect intensity or not, OSS vender always has the most incentive to make its product compatible with its proprietary counterpart, as manifested by the expanded market share after the change. Proprietary software vender, on the other hand, is not willing to make its product compatible with the OSS as doing so will decreases its market share and profit.*

Partially Covered Market with Commerically Supported OSS

In the previous section, the market where open source software and proprietary software battle is assumed to be fully covered. In some situations, however, this condition may not be true, especially when user's learning cost is taken into account. Users may derive negative utilities as software becomes increasingly complicated and requires too much effort to learn, resulting in situations where users are better off not adopting either OSS or proprietary software. This leads to a partially covered market for both software venders.

A commonly observed strategy utilized by OSS vender is to sell a commercially supported version of its open source software. A canonical example would be MySQL, where the software vender offers the MySQL Community Server for free, but charges a $599/server/year license fee for the commercially supported and better performed Enterprise Server version. Compared to its proprietary opponent Oracle, the price MySQL Enterprise version charges for the license fee is much lower. However, it is also believed by many that Oracle still holds a decent lead in product quality and delivers a much better performance than the commercially supported MySQL Enterprise Server. In such a scenario, OSS vender pursues to maximize its profit derived from its commercially supported version rather than reacts passively to the pricing decision of the proprietary software firm. In this section, I extend my model to incorporate the commercially

supported OSS in a partially covered market. It is of most interest to find out how would the impact of network externalities vary and how would software venders be affected regarding their compatibility strategies under the new scenario.

New Benchmark Case – No Network Externalities

Figure 4-2 illustrates the new demands for different software products under the partial market coverage with commercially supported OSS scenario. As usual, I assume consumers choose one and only one preference out of the four options: (1) adopt the free open source software, (2) purchase commercially supported OSS, (3) purchase proprietary software, and (4) do nothing. No matter which product they choose, consumers incur a non-negative cost c in order to get familiar with the functionalities offered by the software. Without network effect, a consumer of type θ derives net utility $u_o = \theta \cdot s_o - c$ by using open source software and $u_{oc} = \theta s_{oc} - p_c - c$ by purchasing commercially supported OSS charging p_c and net utility $u_p = \theta s_p - p_p - c$ by purchasing proprietary software charging p_p. We use oc to denote the open source software with commercial support and p_c , p_p to represent the prices for OC and PS respectively.

In general, one has $s_o < s_{oc} < s_p$ and $p_c < p_p$ to reflect the common practice observed in software market. In addition, in order to guarantee non-negative demand for all three products, we must have the following constraints:

$$p_c > \frac{c\left(s_c - s_o\right)}{s_o}, \qquad p_p > \frac{s_p - s_0}{s_c - s_o} p_c$$

OSS and proprietary software venders then solve the following maximization problems simultaneously:

OSS vender:

$$\max_{p_c} p_c\left(\theta_p - \theta_{oc}\right)$$

$$s.t.\ \frac{c\left(s_c - s_o\right)}{s_o} < p_c < p_p$$

Proprietary software vender:

$$\max_{p_p} p_p\left(1 - \theta_p\right)$$

$$s.t.\ \frac{s_p - s_0}{s_c - s_o} p_c < p_p < p_c + s_p - s_c$$

Accordingly, the optimal prices and profits for OC and PS in the benchmark case are

$p^*_{boc} = \frac{\left(s_c - s_o\right)\left(s_p - s_c\right)}{4s_p - 3s_o - s_c}$, $p^*_{bp} = \frac{2\left(s_p - s_o\right)\left(s_p - s_c\right)}{4s_p - 3s_o - s_c}$ and $\pi^*_{boc} = \frac{\left(s_p - s_o\right)\left(s_c - s_o\right)\left(s_p - s_c\right)}{\left(4s_p - 3s_o - s_c\right)^2}$,

$\pi^*_{bp} = \frac{4\left(s_p - s_o\right)^2\left(s_p - s_c\right)}{\left(4s_p - 3s_o - s_c\right)^2}$. Maximized demands for all three products are given by:

$Q^*_{bo} = \frac{s_p - s_c}{4s_p - 3s_o - s_c} - \frac{c}{s_o}$, $Q^*_{boc} = \frac{s_p - s_o}{4s_p - 3s_o - s_c}$ and $Q^*_{bp} = \frac{2\left(s_p - s_o\right)}{4s_p - 3s_o - s_c}$. Once again, b denotes the benchmark case of no network externalities.

The Impact of Network Externalities on A Partially Covered Market with Commercially Supported OSS

To study the impact of network externalities on a partially covered market with commercially supported OSS, I report the net utilities of open source and proprietary software users and the optimal outcomes under different compatibility strategies and network effect intensities in Table 4-4

When competing software products are compatible

Cases 1 and 2 in Table 4-4 correspond to the situation when all three products are compatible with each other. By setup, all three products share the same network of installed base, from which the network value is generated. Depending on whether open source and proprietary software exhibit the same network effect intensity, network externalities effect exerts different impacts on each product.

Proposition 4-5. *In Case 1 of Table 4-4, network externalities effect has positive impact on the demand of all three products with proprietary software benefits the most followed by the commercial version of OSS and then OSS. Overall profits for both OSS and proprietary software venders increase.*

Proposition 4-5 confirms that positive network externalities increases user's net utility and their willingness to pay, as manifested by the increased overall demand for all three products.

Proposition 4-6. *Define the threshold value* $T_2 = \frac{2s_p^2 - s_o s_p - s_c s_p}{s_c s_p + s_o s_p - 2s_o s_c}$. *In Case 2 of Table 4, if* $\frac{\gamma_o}{\gamma_p} < T_2$, *network externality has positive impact over the demand of proprietary software, open source software and its commercial version, but negative impact over the demand of proprietary software if* $\frac{\gamma_o}{\gamma_p} > T_2$. (Proof is in Appendix)

Threshold value as defined in Proposition 6 can be interpreted as the quality advantage held by the proprietary software. This quality gap can be neutralized when OSS exhibits a higher network effect intensity when $\frac{\gamma_o}{\gamma_p} > T_2$. Once this quality barrier is overcome, network effect starts having a negative impact on proprietary software's demand. This is because the network value generated by OSS offsets its quality disadvantage, and users therefore switch to OSS for higher net utility.

When competing software products are incompatible

Cases 3 and 4 in Table 4-4 correspond to the circumstances when OSS and the proprietary software are incompatible with each other. Note that OSS and its commercial version still share the same network as in essence the two are still the same product and 100% compatible with

each other. I use $Q_{oss} = Q_o + Q_{oc}$ to represent this common installed base. Not surprisingly, the optimal outcomes are too complicated to be compared analytically with the benchmark results. I therefore resort to computational analysis. The three parameters I use in the following numerical analyses are $s_p = 300\%$ of s_o , $s_c = 150\%$ of s_o and $c = 10\%$ of s_o. In addition, I also assume OSS holds a 25% advantage (e.g. $\gamma_o = 125\%$ of γ_p) in network effect intensity when software products do not share the same network effect intensity γ. Choosing $c = 10\%$ of s_o corresponds to a low learning cost so that the effect of c will not overshadow the effect of network externality. The scale and ratio of the parameters are chosen to reflect the relative gap between different variables in the model. For example, we may think of proprietary software's quality is three times as much as that of open source software and twice as much as the commercially supported OSS.

As shown in Figures 4-3 and 4-4, demand curves (solid line) of open source software and its commercial version always stay below the benchmark level demand (dashed line), suggesting network effect has negative impact on their market shares. On the other hand however (see Figure 4-5), the demand curve of proprietary software always stays above the benchmark demand, indicating network effect has positive impact on the market share of proprietary software. The range of network effect intensities $\gamma, \gamma_p \in [0, 0.12]$ is chosen to guarantee non-negative demands for all three products. Note that since $\gamma_o = 125\%$ of γ_p , changing the value of γ_p automatically varies the value of γ_o in Case 4. Also note that demand curve patterns between Case 3 and Case 4 are very similar for all three products, indicating that the difference in network effect intensity values does not significantly alter the impact of network externalities when OSS and proprietary software are incompatible.

In Case 2, quality gaps between OSS and the proprietary software seem to play an important role in deciding the impact of network externalities. In Cases 3 and 4, I find that the reduction of quality difference does help boosting the demand of OSS and lowers the demand of proprietary software slightly. However, it neither drags down the demand curve of proprietary software below the benchmark line nor pulls up the demand curve of OSS above the benchmark line. Same pattern of findings is observed using another set of parameters with significantly reduced quality gaps between all three products. For example set $s_p = 150\%$ of s_o, $s_c = 125\%$ of s_o and $c = 10\%$ of s_o.

In conclusion, I find that when OSS and proprietary software are incompatible with each other, proprietary software product benefits more from the presence of network externalities.

What is the Best Strategy – Compatible or Incompatible?

It is too complicated, if not entirely impossible, to analytically compare the optimal outcomes of all three products before and after they have been made compatible with each other. As a result, I perform numerical analyses using the same set of parameters (as I have used earlier) to investigate which party has the most incentive to make its product compatible with its rival.

By comparing the optimal demands of the three products before and after making them compatible with each other, Figure 4-6, 4-7 and 4-8 illustrate the head to head comparison results of Case 3 vs. 1, and Case 4 vs. 2.

Numerical analyses show that making OSS compatible with the proprietary software is always desirable for the OSS venders. Doing so not only increases the demand of OSS and its commercial version, but also helps OSS impervious to the changes of network effect intensity. For example, as shown in Figures 4-6 and 4-7, when OSS and the proprietary software are

incompatible (solid line), OSS and its commercial version lose their demands to proprietary software as γ increases. However, once they are being made compatible with the proprietary software, increasing network effect intensity benefits both products as evidenced by the upward demand curve (dashed line) in Figures 4-6 and 4-7. Proprietary software venders, on the other hand, are better off without making their products compatible with the OSS. Furthermore, they should try to make their product difficult for the OSS to be compatible with since it is of proprietary software venders' best interest to maintain a market where software products are incompatible with each other. One of the possible strategies to achieve this is by creating new format of files, which can only be accessed by proprietary software product. A good example can be seen in Microsoft Office 2007. Once Microsoft realizes that its OSS opponent – OpenOffice.org™ is fully compatible with the Office 2003 product, they create a new series format of office files (e.g. ".docx" file) in Office 2007 and these formats are found to be difficult to open or edit in OpenOffice.org™ for a long time since the release of Office 2007. Reverse engineering of Office 2007 formats has not been successful thus far, resulting in two incompatible products between OpenOffice.org™ and Office 2007. Not surprisingly, this intentionally created incompatibility has helped Microsoft maintain its dominant position over other productivity software

In summary, in my extended model when a commercially supported OSS is added into the competition and when the market is partially covered due to the non-negligible learning cost, I once again find that OSS venders have the most incentive to make their products compatible with their proprietary opponents but not the other way around. Network externalities have positive impact on proprietary software under most scenarios except for Case 2, where a significantly higher network effect intensity for OSS may allow OSS to take over and win a

decent share of market from its proprietary competitor. Open source software and its commercial version are better off in the presence of network externalities if they are made compatible with proprietary software (e.g. Cases 1 and 2), but worse off in the presence of network externalities if they are not compatible (e.g. Cases 3 and 4).

Summary

I investigate the impact of network externalities on the competition between open source and proprietary software. I analyze four different scenarios depending on whether OSS and proprietary software are compatible with each other and whether they exhibit the same network effect intensities. I first focus on a market that is fully covered. When the market is fully covered, network effect benefits the proprietary software more than OSS in all cases except when both OSS and the proprietary software are compatible with each other and the OSS exhibits stronger network externalities effect. Compared with the case of no network externalities, the installed base and the profit of proprietary software both increase at the expense of OSS losing user base in the presence of network externalities. When both OSS and the proprietary software are compatible with each other and the OSS exhibits stronger network externalities effect, I find that a threshold corresponding to the quality ratio between OSS and proprietary software can be derived such that if the network effect intensity of the OSS is greater than that of the proprietary software multiplied by this threshold value, then OSS benefits from the presence of network externality; otherwise, proprietary software benefits from the presence of network effect.

Next, I relax the condition that the market is fully covered in the model by taking users learning cost into account. In addition, I incorporated the third product, commercially supported OSS, into my model and investigate the impact of network externalities on the competition between proprietary software, OSS and the commercially supported OSS under different scenarios. Analytical and computational results show that network effect benefits the proprietary

software in most cases except for the case where a significantly higher network effect for the OSS may allow OSS to win a decent share of market from its proprietary competitor. Open source software and its commercial version are better off in the presence of network externalities if they are made compatible with proprietary software, but worse off in the presence of network externalities if they are not compatible.

Finally, I find that making software products compatible with competing rival is not desirable by proprietary software vendors but highly favored by OSS venders. Hence, in order to maintain its dominant position in market shares, the proprietary software vendor should make it hard for the OSS software to be compatible with its product.

The analytical insights derived in this paper are based on several premises. First, the OSS and proprietary software under consideration compete for the same market and when software products are compatible, network values derived from OSS network users is the same as the value derived from proprietary software users. Second, I assume that there is no cost difference in using either OSS or proprietary software. To provide more understanding of the competition dynamics between OSS and proprietary software, future research should address each of the foregoing premises. For example, in a unified network (e.g. OSS and proprietary software are compatible), it is quite conceivable that users from OSS network contribute more to a product's value than users from proprietary network. Therefore network values generated by the users from OSS network should be addressed differently than the network values generated by the users from proprietary network. Further, customers' post-purchase learning cost for using OSS may be more than that of proprietary software since OSS does not provide formal technical support, although it is free to adopt OSS initially.

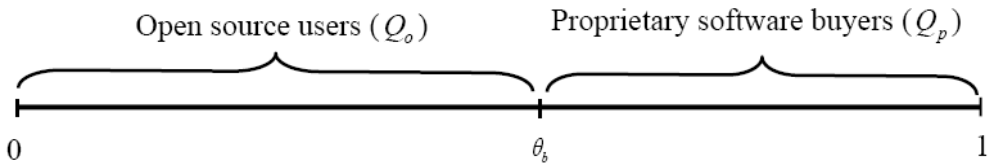


Figure 4-1. Demand of open source and proprietary software

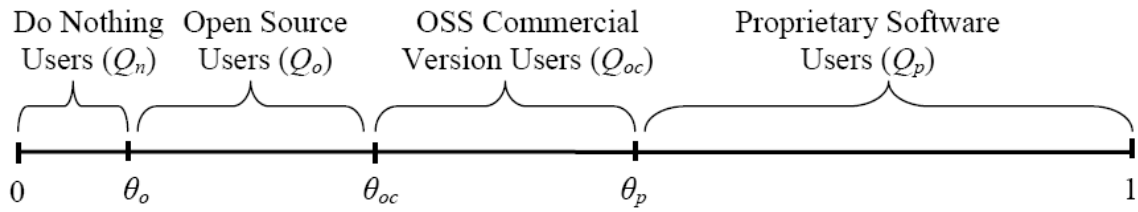
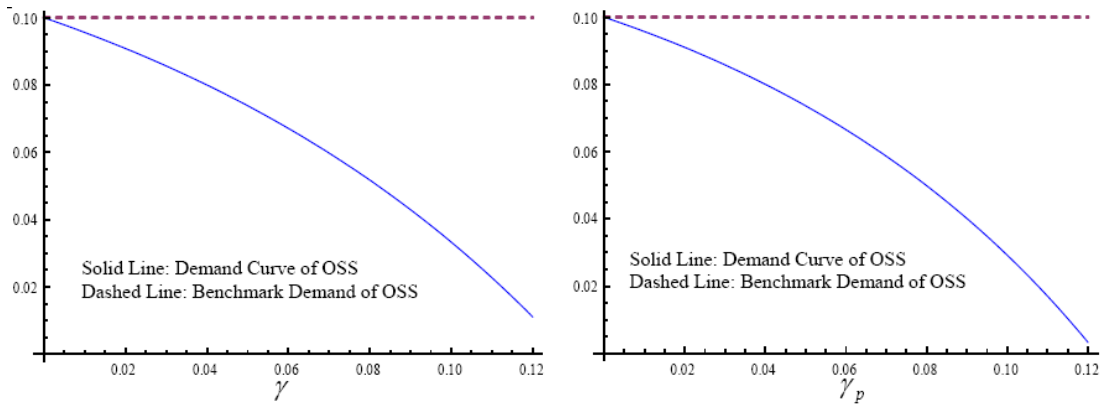
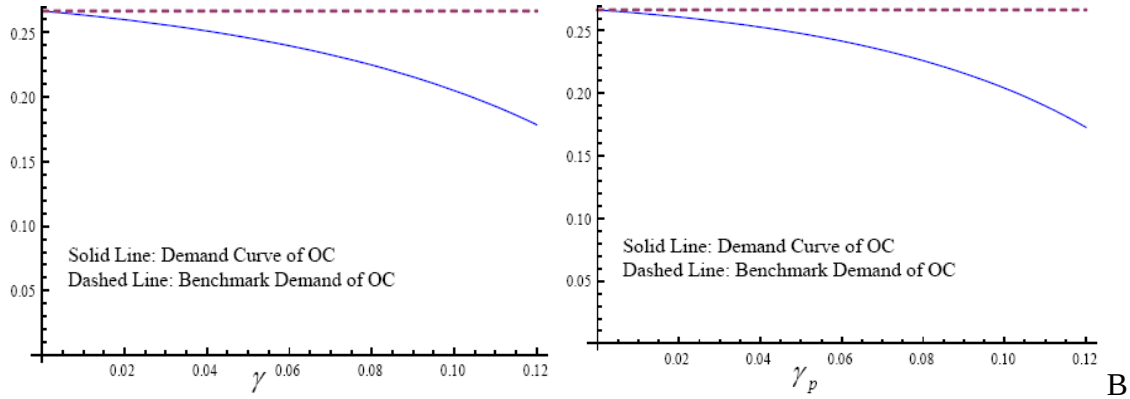


Figure 4-2. Demands of different software products



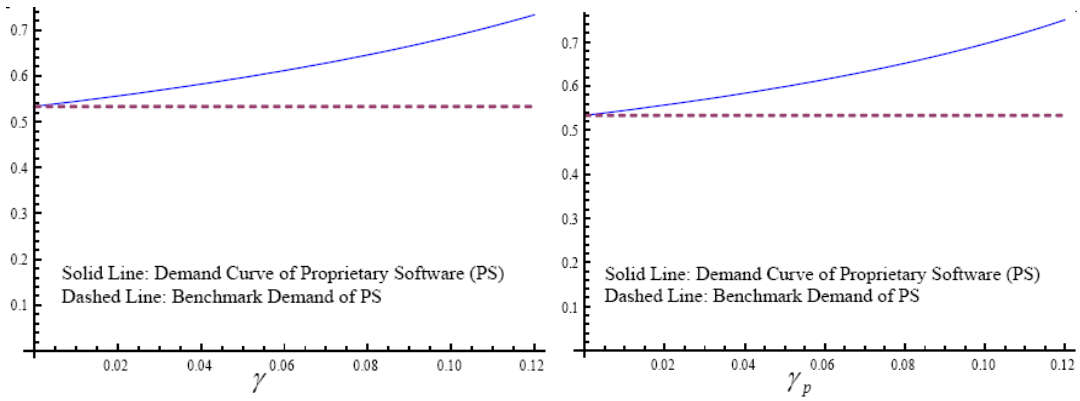
A B

Figure 4-3. Demand curve of OSS w.r.t. network effect intensity. A) Case 3 B) Case 4



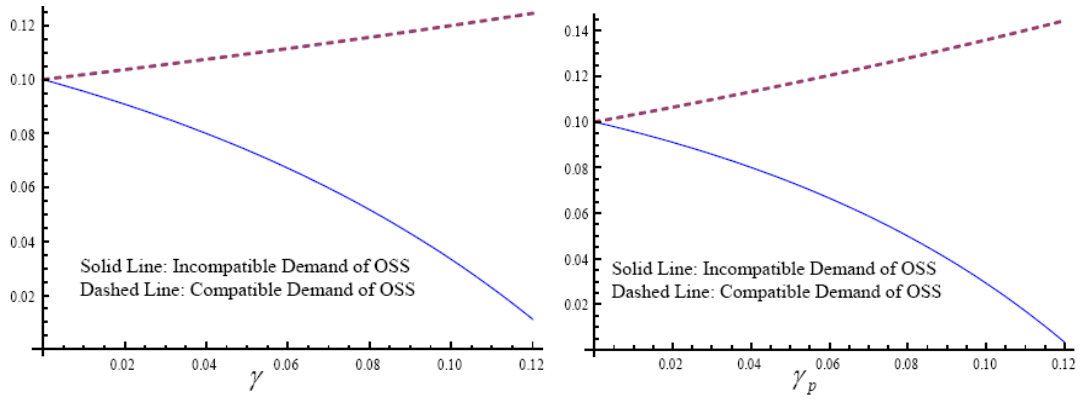
B

Figure 4-4. Demand curve of OC w.r.t. network effect intensity. A) Case 3 B) Case 4



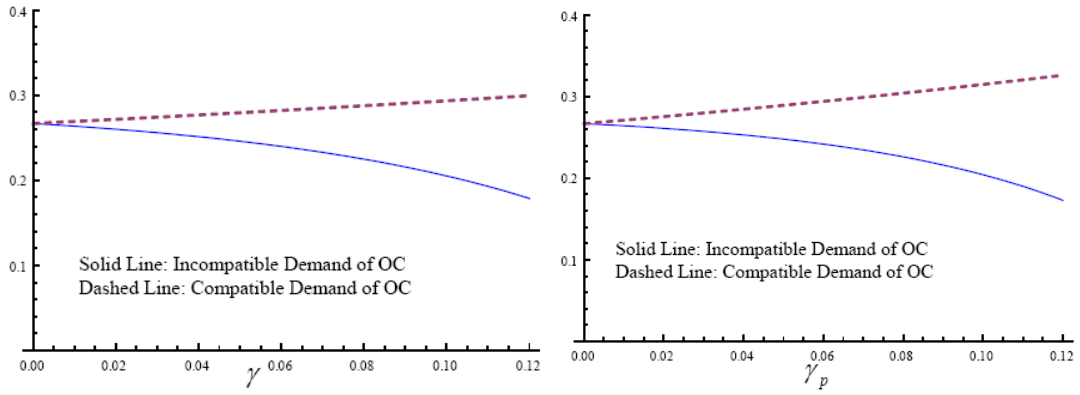
B

Figure 4-5. Demand curve of PS w.r.t. network effect intensity. A) Case 3 B) Case 4



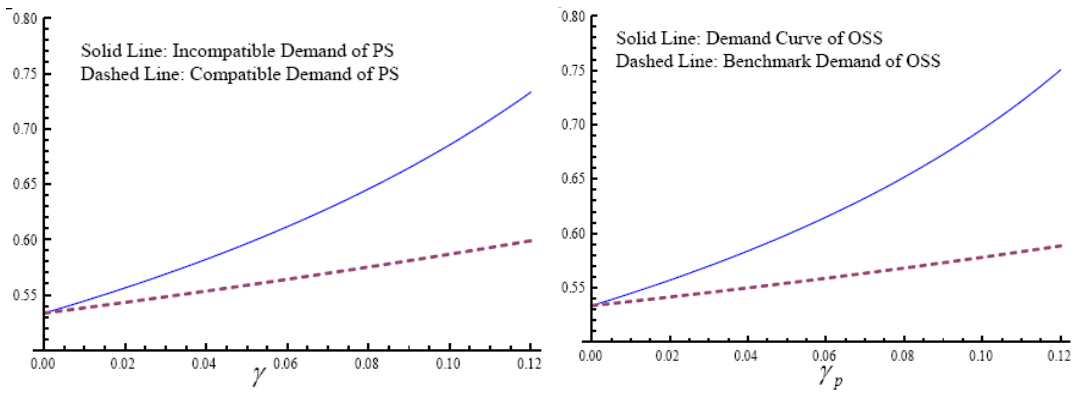
B

Figure 4-6. Optimal demands of OSS. A) Case 1 vs. Case 3 B) Case 2 vs. Case 4



B

Figure 4-7. Optimal demands of OC. A) Case 1 vs. Case 3 B) Case 2 vs. Case 4



B

Figure 4-8. Optimal demands of PS. A) Case 1 vs. Case 3 B) Case 2 vs. Case 4

Table 4-1. Four different cases of analysis

	OSS and Proprietary Software are compatible	OSS and Proprietary Software are incompatible
Both OSS and Proprietary Software exhibit same network effect (γ)	Case 1	Case 3
OSS and Proprietary Software exhibit different Network Effect (γ_o vs. γ_p)	Case 2	Case 4

Table 4-2. Notation of parameters

γ_p: intensity of network externalities of proprietary software

γ_o: intensity of network externalities of open source software (OSS)

θ: consumer type, i.e., consumer's preference or valuation of the quality of the software

s_o: quality of the open source software

s_{oc}: quality of the OSS commercial version

s_p: quality of the proprietary software ($s_p > s_{oc} > s_o$)

Q_o: installed user base of open source software

Q_{oc}: installed user base of open source software's commercial version

Q_p: installed user base of proprietary software

p_p: price of the proprietary software

p_c: price of the open source software's commercial version (OC)

$u = (\theta + \gamma Q)s - p$: Consumer utility function

Table 4-3. Fully covered market: Optimal outcomes

	Case 1	Case 2	Case 3	Case 4
u_o	$(\theta+\gamma Q)s_o$	$(\theta+\gamma_0 Q)s_o$	$(\theta+\gamma Q_o)s_o$	$(\theta+\gamma_o Q_o)s_o$
u_p	$(\theta+\gamma Q)s_p-p$	$u_p=(\theta+\gamma_p Q)s_p-p$	$(\theta+\gamma Q_p)s_p-p$	$(\theta+\gamma_p Q_p)s_p-p$
p^*	$\frac{(s_p-s_o)(1+\gamma)}{2}$	$\frac{s_p(1+\gamma_p)-s_o(1+\gamma_o)}{2}$	$\frac{s_p-s_o(1+\gamma)}{2}$	$\frac{s_p-s_o(1+\gamma_o)}{2}$
Q_o^*	$\frac{(1-\gamma)}{2}$	$\frac{s_p-s_o+s_o\gamma_o-s_p\gamma_p}{2(s_p-s_o)}$	$\frac{s_p-s_o-s_o\gamma-2s_p\gamma}{2(s_p(1-\gamma)-s_o(1+\gamma))}$	$\frac{s_p-s_o-s_o\gamma_o-2s_p\gamma_p}{2(s_p(1-\gamma_p)-s_o(1+\gamma_o))}$
Q_p^*	$\frac{(1+\gamma)}{2}$	$\frac{s_p-s_o+s_p\gamma_p-s_o\gamma_o}{2(s_p-s_o)}$	$\frac{s_p-s_o-s_o\gamma}{2(s_p(1-\gamma)-s_o(1+\gamma))}$	$\frac{s_p-s_o-s_o\gamma_o}{2(s_p(1-\gamma_p)-s_o(1+\gamma_o))}$
π^*	$\frac{(s_p-s_o)(1+\gamma)^2}{4}$	$\frac{(s_o(1+\gamma_o)-s_p(1+\gamma_p))^2}{4(s_p-s_o)}$	$\frac{(s_o-s_p+s_o\gamma)^2}{4(s_p(1-\gamma)-s_o(1+\gamma))}$	$\frac{(s_o-s_p+s_o\gamma_o)^2}{4(s_p(1-\gamma_p)-s_o(1+\gamma_o))}$
Valid γ	$0\le\gamma\le 1$	$0\le\frac{1+\gamma_o}{1+\gamma_p}\le\frac{s_p}{s_o}$	$0\le\gamma\le\frac{s_p-s_o}{2s_p+s_o}$	$0\le\frac{1+\gamma_o}{1-\gamma_p}\le\frac{s_p}{s_o}$

Table 4-4. Partially covered market: Optimal outcomes

	Case 1	Case 2
u_o	$(\theta+\gamma Q)s_o - c$	$(\theta+\gamma_o Q)s_o - c$
u_{oc}	$(\theta+\gamma Q)s_c - p_c - c$	$(\theta+\gamma_o Q)s_c - p_c - c$
u_p	$(\theta+\gamma Q)s_p - p_p - c$	$(\theta+\gamma_p Q)s_p - p_p - c$
p_c^*	$\frac{(s_c-s_o)(s_p-s_c)(s_o-c\gamma)}{s_o(4s_p-3s_o-s_c)(1-\gamma)}$	$-\frac{(s_c-s_o)\left(s_c(s_o-c\gamma_{oss})-s_p\left(s_o(1+\gamma_{oss}-\gamma_p)+c(\gamma_p-2\gamma_{oss})\right)\right)}{s_o(4s_p-3s_o-s_c)(1-\gamma_{oss})}$
p_p^*	$\frac{2(s_p-s_o)(s_p-s_c)(s_o-c\gamma)}{s_o(4s_p-3s_o-s_c)(1-\gamma)}$	$\frac{s_c 2s_o^2+s_o\left(s_p(\gamma_{oss}-\gamma_p-2)+cs_p(\gamma_{oss}+\gamma_p)-2c\gamma_{oss}\right)}{s_o(4s_p-3s_o-sc)(1-\gamma_{oss})}$ $+\frac{+s_c s_p\left(s_o^2(\gamma_{oss}-\gamma_p-2)-2cs_p\gamma_p+s_o\left(2s_p(1-\gamma_{oss}+\gamma_p)+c(\gamma_{oss}+\gamma_p)\right)\right)}{s_o(4s_p-3s_o-sc)(1-\gamma_{oss})}$
Q_o^*	$\frac{(s_p-s_c)(s_o-c\gamma)}{s_o(4s_p-3s_o-s_c)(1-\gamma)}-\frac{c}{s_o}$	$\frac{s_o\left(s_p(1+\gamma_{oss}-\gamma_p)-s_c\right)+c\left(s_c+3s_o(1-\gamma_{oss})+s_p(2\gamma_{oss}+\gamma_p-4)\right)}{s_o(4s_p-3s_o-s_c)(1-\gamma_{oss})}$
Q_{oc}^*	$\frac{(s_p-s_o)(s_o-c\gamma)}{s_o(4s_p-3s_o-s_c)(1-\gamma)}$	$\frac{(s_p-s_o)\left(s_p\left(s_o(1+\gamma_{oss}-\gamma_p)-c(2\gamma_{oss}-\gamma_p)\right)-s_c(s_o-c\gamma_{oss})\right)}{s_o(4s_p-3s_o-s_c)(s_p-s_c)(1-\gamma_{oss})}$
Q_p^*	$\frac{2(s_p-s_o)(s_o-c\gamma)}{s_o(4s_p-3s_o-s_c)(1-\gamma)}$	$\frac{s_c\left(2s_o^2+s_o\left(-2c\gamma_{oss}+s_p(\gamma_{oss}-\gamma_p-2)\right)+cs_p(\gamma_{oss}+\gamma_p)\right)}{s_o(4s_p-3s_o-s_c)(s_p-s_c)(1-\gamma_{oss})}$ $+\frac{+s_p\left(s_o^2(-2+\gamma_{oss}-\gamma_p)-2cs_p\gamma_p+s_o\left(2s_p(1-\gamma_{oss}+\gamma_p)+c(\gamma_{oss}+\gamma_p)\right)\right)}{s_o(4s_p-3s_o-s_c)(s_p-s_c)(1-\gamma_{oss})}$

Table 4-4. (continued)

	Case 3	Case 4
u_o	$\left(\theta+\gamma Q_{oss}\right)s_o-c$	$\left(\theta+\gamma_o Q_{oss}\right)s_o-c$
u_{oc}	$\left(\theta+\gamma Q_{oss}\right)s_c-p_c-c$	$\left(\theta+\gamma_o Q_{oss}\right)s_c-p_c-c$
u_p	$\left(\theta+\gamma Q_p\right)s_p-p_p-c$	$\left(\theta+\gamma_p Q_p\right)s_p-p_p-c$
p_c^*	$\frac{(s_c-s_o)\left(s_p(1-\gamma)\left(s_o-2s_o\gamma-2c(1-\gamma)\gamma\right)-s_c\left(s_o-c\gamma\right)\right)}{s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)(1-\gamma)}$	$\frac{(s_c-s_o)\left(s_p(1-\gamma_{oss})\left(s_o-2s_o\gamma_p-2c(1-\gamma_p)\gamma_{oss}\right)-s_c\left(s_o-c\gamma_{oss}\right)\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)(1-\gamma_{oss})}$
p_p^*	$\frac{s_o s_p(1-\gamma)\left(s_o(-2+\gamma)+(1-\gamma)\left(2s_p(1-\gamma)+c\gamma\right)\right)}{s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)(1-\gamma)}+\frac{s_c\left(2s_o^2+cs_p(1-\gamma)^2\gamma-s_o\left(2c\gamma+s_p\left(2-(3-\gamma)\gamma\right)\right)\right)}{s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)(1-\gamma)}$	$\frac{s_o s_p(1-\gamma_{oss})\left(s_o(-2+\gamma_p)+(1-\gamma_p)\left(2s_p(1-\gamma)+c\gamma_{oss}\right)\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)(1-\gamma_{oss})}$
Q_o^*	$\frac{s_o\left(s_p(1-\gamma)(1-2\gamma)-s_c\right)+c\left(s_c+(1-\gamma)\left(3s_o-2s_p\left(2-(3-\gamma)\gamma\right)\right)\right)}{s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)(1-\gamma)}$	$\frac{s_o\left(s_p(1-\gamma_{oss}-2\gamma_p+2\gamma_p\gamma_{oss})-s_c\right)+c\left(s_c+(1-\gamma_{oss})\left(3s_o-2s_p(2-\gamma_{oss})(1-\gamma_p)\right)\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)(1-\gamma_{oss})}$
Q_{oc}^*	$\frac{\left(s_p(1-\gamma)^2-s_o\right)\left(s_p(1-\gamma)\left(s_o-2s_o\gamma-2c(1-\gamma)\gamma\right)-s_c\left(s_o-c\gamma\right)\right)}{s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)\left(s_p(1-\gamma)^2-s_c\right)(1-\gamma)}$	$\frac{\left(s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-s_o\right)\left(s_p(1-\gamma_{oss})\left(s_o-2s_o\gamma_p-2c(1-\gamma_p)\gamma_{oss}\right)-s_c\left(s_o-c\gamma_{oss}\right)\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)\left(s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-s_c\right)(1-\gamma_{oss})}$
Q_p^*	$\frac{s_o\left(s_p-s_c\right)+cs_c\gamma-s_o s_p\gamma}{3s_o\left(s_p(1-\gamma)^2-s_c\right)}+\frac{s_c\left(s_o-c\gamma\right)+s_o\left(3c\gamma-3s_o+2s_p(1-\gamma)\right)}{3s_o\left(4s_p(1-\gamma)^2-3s_o-s_c\right)}$	$\frac{s_c\left(2s_o^2+s_o\left(-2c\gamma_{oss}-s_p(1-\gamma_{oss})(2-\gamma_p)\right)+cs_p(1-\gamma_p)(1-\gamma_{oss})\gamma_{oss}\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)\left(s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-s_c\right)}+\frac{s_o s_p(1-\gamma_{oss})\left(\left(2s_p(1-\gamma_{oss})+c\gamma_{oss}\right)(1-\gamma_p)-s_o(2-\gamma_p)\right)}{s_o\left(4s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-3s_o-s_c\right)\left(s_p(1-\gamma_p-\gamma_{oss}+\gamma_p\gamma_{oss})-s_c\right)}$

CHAPTER 5
CONCLUSION

In this research, I investigated the optimal software strategies problem in the presence of network externalities – a salient feature commonly observed in a market for widely used computer software. Specifically, I concentrated on three software strategy problems in the presence of network externalities: (1) optimal software free trial strategies; (2) optimal software pricing strategies and (3) optimal compatibility strategies between open source and proprietary software.

In Chapter 2, I provide a thorough analysis about the influence of network effects on the free trial strategies of a software firm under a monopoly scheme. I build analytical models to examine the tradeoff between the effects of reduced uncertainty and demand cannibalization, and uncover the conditions under which software firms should introduce the time-locked free trial. I find that time-locked free trial outperforms the limited version free trial by bringing more profit to the software firm when consumers' prior belief is low and the network effect intensity of the software is moderate.

In Chapter 3, I study the problem of finding the optimal prices of a software product during its life cycle in the presence of coalescing effects of piracy and word-of-mouth. I extend an empirical software diffusion model by converting it from a single period model into a multi-period one. In the settings of only one price change permitted, I show that a market penetration strategy dominates a skimming strategy when the demand of innovators is low, but loses its ground and replaced by the skimming pricing strategy as the demand of innovators increases.

In Chapter 4, I build analytical models to examine the impact of network externalities on the competition between open source software (OSS) and proprietary software. I analyze four different scenarios depending on whether OSS and proprietary software are compatible with each

other and whether they exhibit the same network effect intensities. I show that when the market is fully covered, the installed base and the profit of proprietary software increase at the expense of decreasing user base for OSS in the presence of network externalities. This competitive imbalance becomes more pronounced when a third product, commercially supported OSS, joins the competition. Furthermore, I find that OSS venders always have the most incentive to make its product compatible with proprietary software but not the other way around.

This research work is organized as a collection of articles, each of which corresponds to one chapter of the entire study. Each chapter is complete within itself and includes a conclusion and future work section related with the aspects of the study covered in that specific chapter. Due to this self-contained style of preparation, I ask the readers to refer to those sections for a detailed description of possible future work areas.

APPENDIX
PROOF OF PROPOSITIONS

Proof of Proposition 2-1

Recall that $\pi_{trial}{}^{*} = \dfrac{(1-\tau)(s+\tau\delta-c)^2}{4(s+\tau\delta)(1-\gamma+\gamma\tau)}$ and $\pi_{no-trial}{}^{*} = \dfrac{(s-c)^2}{4s(1-\gamma)}$.

After some algebra, $\pi_{trial}{}^{*} > \pi_{no-trial}{}^{*}$ implies that

$$\gamma(1-\tau)\left[(s-c)^2(s+\tau\delta)-s(s+\tau\delta-c)\right] > (s-c)^2(s+\tau\delta)-(1-\tau)(s+\tau\delta-c)^2 s$$

It can be shown that $(s-c)^2(s+\tau\delta)-s(s+\tau\delta-c)<0$ for $\tau \in [0,1]$.

Therefore $\pi_{trial}{}^{*} > \pi_{no-trial}{}^{*}$ implies that:

$$\gamma < \frac{(s-c)^2(s+\tau\delta)-(1-\tau)(s+\tau\delta-c)^2 s}{(1-\tau)\left[(s-c)^2(s+\tau\delta)-s(s+\tau\delta-c)\right]}$$

Let $\overline{\gamma}_1 = \dfrac{(s-c)^2(s+\tau\delta)-(1-\tau)(s+\tau\delta-c)^2 s}{(1-\tau)\left[(s-c)^2(s+\tau\delta)-s(s+\tau\delta-c)\right]}$

The numerator of $\overline{\gamma}_1$ can be either positive or negative depending on the values of the underlying parameters s, s_t, c, and τ. Since software products exhibit positive network effects (i.e., $\gamma > 0$), $\pi_{trial}{}^{*} > \pi_{no-trial}{}^{*}$ holds when $\overline{\gamma}_1 > 0$ and $\gamma < \overline{\gamma}_1$.

Q.E.D.

Proof of Proposition 2-2

The Lagrangian function of Equation. 2-14 is given by:

$$L(\tau,\lambda) = \frac{(1-\tau)(s+s_t\tau-s\tau-c)^2}{4(s+s_t\tau-s\tau)(1-\gamma+\gamma\tau)} + \lambda(1-\tau)$$

The Kuhn-Tucker conditions require

$$\frac{\partial L}{\partial \tau} = \frac{\begin{array}{c}4[2(1-\tau)(s+\tau\delta-c)\delta-(s+\tau\delta-c)^2](s+\tau\delta)(1-\gamma+\gamma\tau) \\ -4[\delta(1-\gamma+\gamma\tau)+\gamma(s+\tau\delta)](1-\tau)(s+\tau\delta-c)^2\end{array}}{16[(s+\tau\delta)(1-\gamma+\gamma\tau)]^2} - \lambda \le 0 \tag{A-1}$$

$$\tau\frac{\partial L}{\partial \tau} = 0 \tag{A-2}$$

$$\frac{\partial L}{\partial \lambda} = 1-\tau \ge 0 \tag{A-3}$$

$$\lambda\frac{\partial L}{\partial \lambda} = \lambda(1-\tau) = 0 \tag{A-4}$$

$$\lambda \ge 0 \tag{A-5}$$

In Equation A-2, it is obvious that $\tau = 0$ is one of the solutions to equation $\tau\frac{\partial L}{\partial \tau} = 0$. Suppose that $\tau = 0$ is optimal; then it must also satisfy Equation A-1, which implies

$$\frac{4s(1-\gamma)[2(s-c)\delta-(s-c)^2]-4[\delta(1-\gamma)+s\gamma](s-c)^2}{16[s(1-\gamma)]^2} \le 0$$

This in turn implies that: $s(1-\gamma)[2(s-c)\delta-(s-c)^2]-[\delta(1-\gamma)+s\gamma](s-c)^2 \le 0$

After some algebra, one has

$$(s-c)\left[\delta\left(s+c\right)-s\left(s-c\right)\right] \le \delta(s-c)(s+c)\gamma \tag{A-6}$$

From Equation 2-4, we have $0 < P < s+\tau\delta-c$ to ensure $Q_t > 0$. Note that $\tau = 0$ implies $0 < P < s-c$ and $s-c > 0$. Therefore, Inequality A-6 reduces to:

$$\delta\left(s+c\right)-s\left(s-c\right) \le \delta(s+c)\gamma$$

Equivalently, $\gamma \ge 1-\frac{s\left(s-c\right)}{\delta(s+c)}$, since $\delta > 0$ and $s+c$ is always positive.

Therefore $\tau^* = \begin{cases} 0 & \textit{if} \quad \gamma \ge 1-\frac{s\left(s-c\right)}{\delta(s+c)} \\ \varpi & \textit{otherwise} \end{cases}$.

Now let us consider the situation when $\tau = 0$ is not optimal, which implies $\gamma < 1 - \frac{s(s-c)}{\delta(s+c)}$.

In this case, the Kuhn-Tucker condition requires:

$$\frac{\partial L}{\partial \tau} = \frac{\begin{array}{c}[2(1-\tau)(s+\tau\delta-c)\delta-(s+\tau\delta-c)^2](s+\tau\delta)(1-\gamma+\gamma\tau) \\ -[\delta(1-\gamma+\gamma\tau)+\gamma(s+\tau\delta)](1-\tau)(s+\tau\delta-c)^2\end{array}}{4[(s+\tau\delta)(1-\gamma+\gamma\tau)]^2} = 0$$

Let $U = (s+\tau\delta-c)[2(1-\tau)\delta-(s+\tau\delta-c)](s+\tau\delta)(1-\gamma+\gamma\tau)$, and

$$D = [\delta(1-\gamma+\gamma\tau)+\gamma(s+\tau\delta)](1-\tau)(s+\tau\delta-c)^2$$

$\frac{\partial L}{\partial \tau} = 0$ implies that $U = D$.

Therefore the original optimization problem is reduced to find an optimal $\tau \in [0,1]$ that solves the following polynomial.

$$\begin{aligned} f(\tau) &= -\gamma\delta^2\tau^3 + [2\gamma\delta^2 - 2\delta^2 - s\gamma\delta - c\gamma\delta]\tau^2 \\ &+ [\delta^2(1-\gamma) + \delta(2s\gamma + 2c\gamma - 3s)]\tau + [\delta(s - s\gamma - c\gamma) + c\delta - s(s-c)] = 0 \end{aligned}$$

Consequently, one has:

$$f(0) = \delta(s - s\gamma - c\gamma) + c\delta - s(s-c)$$

Since $0 < \gamma < 1 - \frac{s(s-c)}{\delta(s+c)}$, replacing γ by $1 - \frac{s(s-c)}{\delta(s+c)}$ in the above equation leads to

$$f(0) > s(s-c) - c\delta + c\delta - s(s-c) = 0.$$

That is, $f(0) > 0$. Further,

$$f(1) = -(\delta + s)(\delta + s - c)$$

Recall that in order to have $Q_\tau > 0$, we must have $s - c > 0$, which implies $\delta + s - c > 0$.

Therefore one must have $f(1) < 0$.

Since $f(\tau)$ is a well behaved continuous function over the interval $[0,1]$, $f(0)>0$ and $f(1)<0$, there must exist either one or three real roots in $[0,1]$. Moreover, it can be shown that $f(\tau)$ is concave over $[0,1]$ since

$$\begin{aligned}\frac{\partial^2 f(\tau)}{\partial \tau^2} &= -6\gamma\delta^2\tau + 2\left[2\gamma\delta^2 - 2\delta^2 - s\gamma\delta - c\gamma\delta\right] \\ &= \delta\left[-6\gamma\tau\delta + 4\gamma\delta - 4\delta - 2s\gamma - 2c\gamma\right] \\ &= -\delta\left[6\gamma\tau\delta + 4\delta(1-\gamma) + 2s\gamma + 2c\gamma\right] \\ &< 0.\end{aligned}$$

Hence, there exists a unique optimal solution $\varpi \in [0,1]$ such that $f(\varpi)=0$.

Q.E.D.

Proof of Proposition 2-3

$\pi_{trial}{}^* = \dfrac{(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2}{4(s+\tau\delta)\left(1-\gamma(1-\tau)(1+\rho\tau)\right)}$ and $\pi_{no-trial}{}^* = \dfrac{(s-c)^2}{4s(1-\gamma)}$.

$\pi_{trial}{}^* > \pi_{no-trial}{}^*$ implies that

$$\frac{(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2}{4(s+\tau\delta)\left(1-\gamma(1-\tau)(1+\rho\tau)\right)} > \frac{(s-c)^2}{4s(1-\gamma)}$$

$L.H.S$

$(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s(1-\gamma)$

$= (1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s - \gamma(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s$

$R.H.S.$

$(s-c)^2(s+\tau\delta)\left(1-\gamma(1-\tau)(1+\rho\tau)\right)$

$= (s-c)^2(s+\tau\delta) - \gamma(s-c)^2(s+\tau\delta)(1-\tau)(1+\rho\tau)$

$L.H.S > R.H.S$

$\Rightarrow \gamma(s-c)^2(s+\tau\delta)(1-\tau)(1+\rho\tau) - \gamma(1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s > (s-c)^2(s+\tau\delta) - (1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s$

$\Rightarrow \gamma(1-\tau)(1+\rho\tau)\left[(s-c)^2(s+\tau\delta) - (s+\tau\delta-c)^2 s\right] > (s-c)^2(s+\tau\delta) - (1-\tau)(1+\rho\tau)(s+\tau\delta-c)^2 s$

After some algebra it can be shown that:

$(s-c)^2(s+\tau\delta)-(s+\tau\delta-c)^2 s<0$ for any $\tau\in[0,1]$.

Let $\overline{\gamma}_3=\dfrac{(s-c)^2(s+\tau\delta)-(1-\tau)\left(1+\rho\tau\right)(s+\tau\delta-c)^2 s}{(1-\tau)\left(1+\rho\tau\right)\left[(s-c)^2(s+\tau\delta)-(s+\tau\delta-c)^2 s\right]}$.

The numerator of $\overline{\gamma}_3$ can be positive or negative depending on the values of the underlying parameters s, s_t, c, and τ. Since software products exhibit positive network effects (i.e., $\gamma>0$), $\pi_{trial}{}^*>\pi_{no-trial}{}^*$ holds when $\overline{\gamma}_3>0$ and $\gamma<\overline{\gamma}_3$.

Q.E.D.

Proof of Proposition 2-4

Software firm seeks to jointly decide the optimal free trial time and the optimal price of its commercial product so that its profit is maximized as below:

$$\max_{\tau,\ P}\pi=PQ_\tau=Q_\tau=\frac{\left(1-\tau\right)\left(1+\rho\tau\right)\left(s+\tau\delta-P-c\right)}{\left(s+\tau\delta\right)\left(1-\gamma\left(1-\tau\right)\left(1+\rho\tau\right)\right)}P$$

s.t. $0\le P\le s+\tau\delta-c$, and $0\le\tau\le 1$

To solve this problem, I first treat the free trial time τ as given, and the problem reduces to the fixed trial time problem, where the optimal price and the overall profit as a function of τ are given by: $P^*(\tau)=\dfrac{s+\tau\delta-c}{2}$, and $\pi^*(\tau)=\dfrac{(1-\tau)\left(1+\rho\tau\right)(s+\tau\delta-c)^2}{4(s+\tau\delta)\left(1-\gamma\left(1-\tau\right)\left(1+\rho\tau\right)\right)}$ respectively. Then, I evaluate the change of $\pi^*(\tau)$ with respect to τ:

$$\max_{\tau}\pi^*(\tau)=\frac{(1-\tau)\left(1+\rho\tau\right)(s+\tau\delta-c)^2}{4(s+\tau\delta)\left(1-\gamma\left(1-\tau\right)\left(1+\rho\tau\right)\right)} \tag{A-7}$$

s.t. $0\le\tau\le 1$

The Lagrangian function of Equation A-7 is given by:

$$L(\tau,\lambda)=\frac{(1-\tau)\left(1+\frac{1}{2}a\tau\right)(s+s_t\tau-s\tau-c)^2}{4(s+s_t\tau-s\tau)\left(1-\gamma\left(1-\tau\right)\left(1+\frac{1}{2}a\tau\right)\right)}+\lambda(1-\tau)$$

The Kuhn-Tucker conditions require

$$\frac{\partial L}{\partial \tau}=\frac{\left[4(1-\tau)\left(1+\rho\tau\right)(s+\tau\delta-c)(s_t-s)+\left(a\left(1-\tau\right)-2\left(1+\rho\tau\right)\right)(s+\tau\delta-c)^2\right]\left(s+\tau\delta\right)\left(1-\gamma\left(1-\tau\right)\left(1+\rho\tau\right)\right)}{8\left[(s+\tau\delta)\left(1-\gamma\left(1-\tau\right)\left(1+\rho\tau\right)\right)\right]^2}-\lambda\le 0 \qquad \text{(A-8)}$$

$$\tau\frac{\partial L}{\partial \tau}=0 \qquad \text{(A-9)}$$

$$\frac{\partial L}{\partial \lambda}=1-\tau\ge 0 \qquad \text{(A-10)}$$

$$\lambda\frac{\partial L}{\partial \lambda}=\lambda(1-\tau)=0 \qquad \text{(A-11)}$$

$$\lambda\ge 0 \qquad \text{(A-12)}$$

In Equation A-9, it is obvious that $\tau=0$ is one of the solutions to equation $\tau\frac{\partial L}{\partial \tau}=0$.

Suppose that $\tau=0$ is optimal; then it must also satisfy inequality A-8, which implies:

$$\frac{s(1-\gamma)\left(\left(\rho-1\right)(s-c)^2+4\left(s-c\right)\delta\right)-(s-c)^2\left(\left(\rho-a\right)s\gamma+2\left(1-\gamma\right)\delta\right)}{4[s(1-\gamma)]^2}\le 0$$

This in turn implies that: $s\left(2\rho\left(c-s\right)+4s-2\left(s+c\right)\gamma\right)-2\left(s+c\right)\left(1-\gamma\right)\left(s+\delta\right)\ge 0 \quad \& \quad \gamma\ne 1$

After some algebra, one has: $2\gamma\left(s+c\right)\delta\ge 2\left(s+\delta\right)\left(s+c\right)-4s^2+2\rho s\left(s-c\right)$ (A-13)

Or equivalently, $\gamma\ge\frac{\left(s+\delta\right)\left(s+c\right)-2s^2+\rho s\left(s-c\right)}{\left(s+c\right)\delta}$

Therefore $\tau^*=0$ if $\gamma\ge\frac{\left(s+\delta\right)\left(s+c\right)-2s^2+\rho s\left(s-c\right)}{\left(s+c\right)\delta}$

Q.E.D

Proof of Proposition 4-1

Define Δ as the difference when comparing the optimal solutions with benchmark level results

When the market is fully covered, in case 1, we have:

$$\Delta_o = Q_o^* - Q_{bo}^* = -\frac{\gamma}{2} < 0$$
$$\Delta_p = Q_p^* - Q_{bp}^* = \frac{\gamma}{2} > 0$$
$$\Delta_{price} = p^* - p_b^* = \frac{\left(s_p - s_o\right)\gamma}{2} > 0$$
$$\Delta_\pi = \pi^* - \pi_b^* = \frac{\left(s_p - s_o\right)}{4}\left(\left(1+\gamma\right)^2 - 1\right) > 0$$

Q.E.D.

Proof of Proposition 4-2

Case 3 and 4 correspond to the scenario when OSS and proprietary software are incompatible. We use Q_{io} to denote the optimal demand for open source software under case i, and Q_{ip} to denote the optimal demand for proprietary software under case i.

<u>Case 3:</u>

In order to guarantee non-negative demands for both software products, we must have:

$$\theta_{3o} = \frac{\left(s_o + s_p\right)\gamma Q_o - s_p\gamma + p}{s_p - s_o} = \frac{p - s_p\gamma}{s_p\left(1-\gamma\right) - s_o\left(1+\gamma\right)} \geq 0 \qquad \text{(A-14)}$$

which in turn requires:

$$p - s_p\gamma \geq 0$$
$$s_p\left(1-\gamma\right) - s_o\left(1+\gamma\right) \geq 0$$

It is worth noting that when both inequalities above are the other way around, the condition of positive demand is still satisfied. However, this in turn renders a positive relationship between the price and the demand. Obviously, this result contradicts the law of

demand where the higher the price, the lower the quantity demanded. As a result, we require the constraint for positive demand be the format as given above.

Also since the entire market size is normalized to 1, we must have:

$$\theta_{3o} = \frac{p - s_p\gamma}{s_p(1-\gamma) - s_o(1+\gamma)} \le 1 \tag{A-15}$$

Solving inequalities A-14 and A-15 together, we find the valid range of γ as the condition given below:

$$0 \le \gamma \le \frac{s_p - s_o}{2s_p + s_o}$$

Comparing the optimal solutions with benchmark level results assuming γ is chosen within its valid range, we have:

$$p^* = \frac{s_p - s_o(1+\gamma)}{2} < \frac{s_p - s_o}{2} = p_b^*$$
$$Q_{3o}^* = \frac{s_p - s_o - s_o\gamma - 2s_p\gamma}{2\left(s_p(1-\gamma) - s_o(1+\gamma)\right)} < \frac{s_p - s_o - s_o\gamma}{2\left(s_p(1-\gamma) - s_o(1+\gamma)\right)} = Q_{3p}^*$$

Recall that in benchmark case we have: $Q_o^* = Q_p^* = \frac{1}{2}$. Under case 3, since $Q_{3o}^* < Q_{3p}^*$ and the market is fully covered, the demand for proprietary software therefore increases and the demand of open source software decreases.

The profit of proprietary software increases when compared to the benchmark level profit as below:

$$\Delta_\pi = \pi_3^* - \pi_b^* = \frac{\gamma\left(\left(s_p - s_o\right)^2 + s_o^2\gamma\right)}{4\left(s_p - s_o - \left(s_o + s_p\right)\gamma\right)} \ge 0 \qquad \text{when } 0 \le \gamma \le \frac{s_p - s_o}{2s_p + s_o}$$

<u>Case 4:</u>

Similar procedures can be applied when OSS displays a higher network effect intensity value in case 4.

To guarantee non-negative demands, we must have:

$$0 \le \frac{1+\gamma_o}{1-\gamma_p} \le \frac{s_p}{s_o}$$

Comparing the optimal solutions with benchmark level results assuming γ_o, γ_p are chosen within their valid ranges, we have:

$$p^* = \frac{s_p - s_o\left(1+\gamma_o\right)}{2} < \frac{s_p - s_o}{2} = p_b^*$$

$$Q_{4o}^* = \frac{s_p - s_o - s_o\gamma_o - 2s_p\gamma_p}{2\left(s_p\left(1-\gamma_p\right) - s_o\left(1+\gamma_o\right)\right)} < \frac{s_p - s_o - s_o\gamma_o}{2\left(s_p\left(1-\gamma_p\right) - s_o\left(1+\gamma_o\right)\right)} = Q_{4p}^*$$

For the same reason as in case 3, the demand for proprietary software increases and the demand of open source software decreases in case 4.

The profit of proprietary software increases when compared to the benchmark level profit as below:

$$\Delta_\pi = \pi_4^* - \pi_b^* = \frac{1}{4}\left(s_p - s_o + \frac{\left(s_o - s_p + s_o\gamma_o\right)^2}{s_p\left(1-\gamma_p\right) - s_o\left(1+\gamma_o\right)}\right) \ge 0 \qquad \text{when } 0 \le \frac{1+\gamma_o}{1-\gamma_p} \le \frac{s_p}{s_o}$$

Q.E.D.

Proof of Proposition 4-3

Under case 2, OSS and proprietary are compatible with each other but display different network effect intensity values.

Comparing the optimal solutions with bench mark level results, we have:

$$\Delta_o = Q_o^* - Q_{bo}^* = \frac{s_o\gamma_o - s_p\gamma_p}{2\left(s_p - s_o\right)}$$

$$\Delta_p = Q_p^* - Q_{bp}^* = \frac{s_p\gamma_p - s_o\gamma_o}{2\left(s_p - s_o\right)}$$

A threshold value can be defined, such that:

$$\begin{cases} \Delta_o > 0,\ \Delta_p < 0 & \text{if } \dfrac{\gamma_o}{\gamma_p} > \dfrac{s_p}{s_o} \\ \Delta_o < 0,\ \Delta_p > 0 & \text{Otherwise} \end{cases}$$

If the ratio between two network effects ($\frac{\gamma_o}{\gamma_p}$) is greater than the threshold value ($\frac{s_p}{s_o}$), then open source software benefits from the network effect, otherwise proprietary software benefits from the network effect.

The change of optimal price and profit for proprietary software also depend on this threshold value:

$$p^* = \begin{cases} \dfrac{s_p\left(1+\gamma_p\right) - s_o\left(1+\gamma_o\right)}{2} < \dfrac{s_p - s_o}{2} = p_b^* & \text{if } \dfrac{\gamma_o}{\gamma_p} > \dfrac{s_p}{s_o} \\ \dfrac{s_p\left(1+\gamma_p\right) - s_o\left(1+\gamma_o\right)}{2} > \dfrac{s_p - s_o}{2} = p_b^* & \text{Otherwise} \end{cases}$$

$$\pi^* = \begin{cases} \dfrac{\left(s_o\left(1+\gamma_o\right) - s_p\left(1+\gamma_p\right)\right)^2}{4\left(s_p - s_o\right)} < \dfrac{s_p - s_o}{4} = \pi_b^* & \text{if } \dfrac{\gamma_o}{\gamma_p} > \dfrac{s_p}{s_o} \\ \dfrac{\left(s_o\left(1+\gamma_o\right) - s_p\left(1+\gamma_p\right)\right)^2}{4\left(s_p - s_o\right)} > \dfrac{s_p - s_o}{4} = \pi_b^* & \text{Otherwise} \end{cases}$$

Q.E.D.

Proof of Proposition 4-4

Suppose γ is chosen within its valid range

<u>Case 1 vs. Case 3:</u>

$$Q_{1o}^{*}-Q_{3o}^{*}=\frac{\gamma\left(s_o+\left(s_o+s_p\right)\gamma\right)}{2\left(s_p\left(1-\gamma\right)-s_o\left(1+\gamma\right)\right)}>0\Rightarrow Q_{1o}^{*}>Q_{3o}^{*}$$

$$Q_{1p}^{*}-Q_{3p}^{*}=-\frac{\gamma\left(s_o+\left(s_o+s_p\right)\gamma\right)}{2\left(s_p\left(1-\gamma\right)-s_o\left(1+\gamma\right)\right)}<0\Rightarrow Q_{1p}^{*}<Q_{3p}^{*}$$

$$\pi_1^{*}-\pi_3^{*}=\frac{1}{4}\left(-\left(s_p-s_o\right)\left(1+\gamma\right)^2-\frac{(s_o-s_p+s_o\gamma)^2}{s_p\left(1-\gamma\right)-s_o\left(1+\gamma\right)}\right)<0\Rightarrow \pi_1^{*}<\pi_3^{*}$$

<u>Case 2 vs. Case 4:</u>

First let us assume $\frac{\gamma_o}{\gamma_p}>\frac{s_p}{s_o}$

From proposition 4-3 we know that OSS benefits from the presence of network externality under case 2, which implies: $Q_{2o}^{*}>\frac{1}{2}$ and $Q_{2p}^{*}<\frac{1}{2}$. Also from proposition 4-2 we know that proprietary benefits from the presence of network externality under case 4, which implies $Q_{4o}^{*}<\frac{1}{2}$ and $Q_{4p}^{*}>\frac{1}{2}$. Therefore we have: $Q_{2o}^{*}>\frac{1}{2}>Q_{4o}^{*}$ and $Q_{2p}^{*}<\frac{1}{2}<Q_{4p}^{*}$.

Since $Q_{2o}^{*}>Q_{4o}^{*}$, OSS vender has the most incentive to make its product compatible.

Now let us consider the case when $\frac{\gamma_o}{\gamma_p}<\frac{s_p}{s_o}$

$$\frac{\gamma_o}{\gamma_p}<\frac{s_p}{s_o}\Rightarrow s_p\gamma_p>s_o\gamma_o\Rightarrow s_p^2\gamma_p^2>s_o^2\gamma_o^2$$

$$Q_{2o}^{*}-Q_{4o}^{*}=\frac{s_o\gamma_o\left(s_p-s_o\right)+\left(s_p^2\gamma_p^2-s_o^2\gamma_o^2\right)}{2\left(s_p-s_o\right)\left(s_p\left(1-\gamma_p\right)-s_o\left(1+\gamma_o\right)\right)}>0\Rightarrow Q_{2o}^{*}>Q_{4o}^{*}$$

$$Q_{2p}^{*}-Q_{4p}^{*}=\frac{-\left(s_p^2\gamma_p^2-s_o^2\gamma_o^2\right)-s_o\gamma_o\left(s_p-s_o\right)}{2\left(s_p-s_o\right)\left(s_p\left(1-\gamma_p\right)-s_o\left(1+\gamma_o\right)\right)}<0\Rightarrow Q_{2p}^{*}<Q_{2p}^{*}$$

Once again, we find OSS vender has the most incentive to make its product compatible. The demand for proprietary software however decreases under both conditions if the products

are made from incompatible to compatible. In addition, the following equation shows that the profit for proprietary software also drops.

$$\pi_2^* - \pi_4^* = \frac{1}{4}\left(\frac{(s_o - s_p + s_o\gamma_o)^2}{s_p\left(1-\gamma_p\right) - s_o\left(1+\gamma_o\right)} + \frac{\left(s_p\left(1-\gamma_p\right) - s_o\left(1+\gamma_o\right)\right)^2}{s_p - s_o} \right) > 0 \Rightarrow \pi_2^* < \pi_4^*$$

Q.E.D.

Proof of Proposition 4-5

Compare to benchmark level results under extended model:

$$p_{1c}^* - p_{bc}^* = \frac{\left(s_o - c\right)\left(s_c - s_o\right)\left(s_p - s_c\right)\gamma}{s_o\left(4s_p - 3s_o - s_c\right)\left(1-\gamma\right)} > 0$$

$$p_{1c}^* - p_{bp}^* = \frac{2\left(s_o - c\right)\left(s_p - s_o\right)\left(s_p - s_c\right)\gamma}{s_o\left(4s_p - 3s_o - s_c\right)\left(1-\gamma\right)} > 0$$

$$\Delta_{1o} = Q_{1o}^* - Q_{bo}^* = \frac{\left(s_p - s_c\right)\left(s_o - c\right)\gamma}{s_o\left(4s_p - 3s_o - s_c\right)\left(1-\gamma\right)} > 0$$

$$\Delta_{1oc} = Q_{1oc}^* - Q_{boc}^* = \frac{\left(s_p - s_o\right)\left(s_o - c\right)\gamma}{s_o\left(4s_p - 3s_o - s_c\right)\left(1-\gamma\right)} > 0$$

$$\Delta_{1p} = Q_{1p}^* - Q_{bp}^* = \frac{2\left(s_p - s_o\right)\left(s_o - c\right)\gamma}{s_o\left(4s_p - 3s_o - s_c\right)\left(1-\gamma\right)} > 0$$

All three products benefit from the presence of network externalities with proprietary software benefit the most, followed by the commercial version of the OSS and then OSS. As seen in: $\Delta_{1o} < \Delta_{1oc} < \Delta_{1p}$

Overall profits for proprietary software and OSS venders both increases since $\pi_c^* = p_{1c}^* Q_{1oc}^* > p_{bc}^* Q_{boc}^* = \pi_{bc}^*$ and $\pi_p^* = p_{1p}^* Q_{1p}^* > p_{bp}^* Q_{bp}^* = \pi_{bp}^*$.

Q.E.D.

Proof of Proposition 4-6

Compare to benchmark level results under extended model:

$$\Delta_{2o} = Q_{2o}^{*} - Q_{bo}^{*} = \frac{(s_o - c)\left((s_p - s_c)\gamma_o + (\gamma_o - \gamma_p)s_p\right)}{s_o(4s_p - 3s_o - s_c)(1-\gamma_o)} > 0$$

$$\Delta_{2oc} = Q_{2oc}^{*} - Q_{boc}^{*} = \frac{(s_o - c)(s_p - s_o)\left((s_p - s_c)\gamma_o + (\gamma_o - \gamma_p)s_p\right)}{s_o(4s_p - 3s_o - s_c)(s_p - s_c)(1-\gamma_o)} > 0$$

$$\Delta_{2p} = Q_{2p}^{*} - Q_{bp}^{*} = \frac{(s_o - c)\left(\gamma_p\left(2s_p^2 - s_o s_p - s_c s_p\right) + \gamma_{o\,ss}\left(2s_o s_c - s_c s_p - s_o s_p\right)\right)}{s_o(4s_p - 3s_o - s_c)(s_p - s_c)(1-\gamma_{oss})}$$

A threshold value can be derived such that:

$$\begin{cases} \Delta_{2p} < 0 & if \ \dfrac{\gamma_o}{\gamma_p} > \dfrac{2s_p^2 - s_o s_p - s_c s_p}{s_c s_p + s_o s_p - 2s_o s_c} \\ \Delta_{2p} > 0 & \text{Otherwise} \end{cases}$$

Under Case 2 only open source software benefit from the presence of network externalities. Depending on the underlying parameter, the impact for the network effect can be negative for proprietary software.

Q.E.D.

LIST OF REFERENCES

American Electronic Association. Cyberstates 2001. A state-by-state overview of the high-technology industry by AeA.

American Electronic Association. 2002. Tech employment update.

Asundi, J., O. Carare, K. Dogan. 2008. Competitive implications of software open-sourcing. Working paper, University of Texas at Dallas. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1185374.

Bawa, K., R. Shoemanker. 2004. Effects of free sample promotions on incremental brand sales. *Marketing Science*. **23** 345–363.

Bass, F., M. 1969. A new product growth model for consumer durables. *Management Science*. **15** 215–227.

Bergemann, D., K. H. Schlag. 2007. Pricing without priors. Working paper, Cowles Foundation for Research in Economics at Yale University. http://ssrn.com/abstract=1013570.

Berst, J. 2002. Why technology can't stop music piracy. *ZDNET Anchor Desk*. Available at: http://www.zdnet.com/anchordesk/stories/story/0,10738,2677668,00.html

Bettinger, C.O., D. E. Lydon, W. G. Hugh. 1979. The impact of free-sample advertising. *Journal of Advertising Research*. **19** 35–40.

Black, J. Nov. 4th, 2003. The penguin's new desktop habitat. *BusinessWeek*.

Business Software Alliance. 2007. Global software piracy report. Available at: http://w3.bsa.org/globalstudy/

Cabral, L. M. B., D. J. Salant, G. A. Woroch. 1999. Monopoly pricing with network externalities. *International Journal of Industrial Organization*. **17**(2) 199–214.

Casadesus-Masanell, R., P. Chemawat. 2006. Dynamic mixed duopoly: a model motivated by Linux vs. Windows. *Management Science*. **52**(7) 1072–1084

Chellappa, R. K., S. Shivendu. 2005. Managing piracy: pricing and sampling strategies for digital experience goods in vertically segmented markets. *Information Systems Research*. **16**(4) 400–417.

Chen, H.G., H. K. Cheng, P. Chin, G. J. Koehler, F. R. Lin, S. M. Lin. 2005 An agent based simulation of competition between open source and commercial software. Working paper, Warrington College of Business Administration at the University of Florida.

Chen, H.G., H. K. Cheng, H. T. Hsu, G. J. Koehler. 2005. The diffusion of open source and commercial web servers: an empirical study. Working paper, Warrington College of Business Administration at the University of Florida

Chen, Y., I. P. L. Png. 1999. Software pricing and copyright enforcement: private profit vis-à-vis social welfare. *Proceedings of the 20th International Conference on Information Systems (ICIS)*. 119–123.

Chen, Y., I. Png. 2003. Information goods pricing and copyright enforcement: welfare analysis. *Information Systems Research*. **14** 107–123

Cheng, H. K., Q. Tang. 2007. Free trial or no free trial: optimal software product design with network effects. Working Paper, Social Science Research Network. http://ssrn.com/abstract=1082120.

Cheng, H. K., R., R. Sims, H. Teegen. 1997. To purchase or to pirate software: an empirical study. *Journal of Management Information Systems* **13**. 49–60.

Chengalur, S. S., A. Sidorova. 2003. Survival of open-source projects: a popular ecology perspective. *Proceedings of the International Conference on Information Systems*. 782–787.

Church, J., N. Gandal, D. Krause. 2003. Indirect network effects and adoption externalities. *CEPR* discussion paper No. 3738. http://ssrn.com/abstract=383802

Conner, K. 1995. Obtaining strategic advantage from being imitated: when can encouraging 'clones' pay. *Management Science*. **41**(2) 209–225.

Conner, K., R.P. Rumelt. 1991. Software piracy: An analysis of protection strategies, *Management Science*. **37**(2) 125–139.

Crowston, K., H. Kripalani, J. Green. 2003. Defining open source software project success. *Proceedings of the International Conference on Information Systems*. 329–340.

Datta, D. Jan. 13th , 2003. Open source software to gain currency: PWC. *BusinessLine*.

de Freitas, D. 1994. *The Fight Against Piracy*. International Publishers Association, Paris, France.

Drake, M. 2003. *Encyclopedia of Library and Information Science*. CRC Press.

Economides, N. 1996. Network externalities, complementarities, and invitations to enter. *European Journal of Political Economy*. **12**(2) 211–233.

Einhor, B., M. Kripalani, J. Green. Jan. 12th, 2004 Asia loves Linux – and Microsoft scrambles. *Businessweek*.

Ellison, G., D. Fudenberg. 1995. Word-of-mouth communication and social learning. *The Quarterly Journal of Economics*. **110**(1) 93–125.

Farrell, J., G. Saloner. 1986. Installed base and compatibility: innovation, product preannouncements and predation. *American Economic Review*. **76**(5) 940–955.

Fitzgerald, B., J. Feller. 2001. Open source software: investigating the software engineering, psychosocial and economic issue. *Information Systems Journal*. **11**(4) 273–276.

Fitzgerald, B., T. Kenny. 2003 Open source software in the trenches: lessons from a large-scale OSS implementation. *Proceedings of the International Conference on Information Systems*. 316–326.

Friedmand, M. A., J. M. Voas. 1995 *Software assessment – reliability, safety, testability*. John Wiley & Sons, Inc., New York.

Givon, M., V. Mahajan, E. Muller. 1995. Software piracy: estimation of lost sales and the impact on software diffusion. *Journal of Marketing*. **59** 29–37.

Godenberg, J., B. Liba, E. Muller. 2001. Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Marketing Letters*. **12**(3) 211–223

Gopal, R. D., G.L. Sanders. 1997. Preventive and deterrent controls for software piracy. *Journal of Management Information Systems*. **13**(44) 29–48.

Gopal, R. D., G.L. Sanders.1998. International software piracy: Analysis of key issues and impacts. *Information Systems Research*. **9**(4) 380–397.

Gopal, R. D., G.L. Sanders. 2000. Global software piracy: You can't get blood out of a turnip. *Communications of the ACM*. **43**(9) 82–89.

Hann, I. H., J. Roberts, S. A. Slaughters, R. Fielding. 2002. Economic incentives for participate in open source software projects. *Proceedings of the International Conference on Information Systems*. 365–372

Hann, I. H., J. Roberts, S. A. Slaughters. 2004 Why developers participate in open source software projects: an empirical investigation. *Proceedings of the International Conference on Information Systems*. 821–830.

Heiman, A., E. Muller. 1996. Using demonstration to increase new product acceptance: controlling demonstration time. *Journal of Market Research*. **33**(4) 422–430.

Heiman, A., B. McWilliams, Z. Shen, D. Zilberman. 2001. Learning and forgetting: modeling optimal product sampling over time. *Management Science*. **47** 532–546.

Jain, D., V. Mahajan, E. Muller. 1995. An approach for determining optimal product sampling for diffusion of a new product. *Journal of Product Innovation Management*. **12** 124–135.

Johnson, W. 1985. The economics of copying. *Journal of Political Economy*. **93** 158–173.

Katz, M., C. Shapiro. 1985. Network externalities, competition and compatibility. *American Economic Review*. **75**(3) 424–440.

Katz, M., C. Shapiro. 1994. Systems competition and network effects. *The Journal of Economic Perspectives*. **8** 93-115.

Kempf, D., R. E. Smith. 1998. Consumer processing of product trial and the influence of prior advertising: a structural modeling approach. *Journal of Marketing Research*. **35** 325–338.

Khouja, M., S. Park. 2007. Optimal pricing of digital experience goods under piracy. *Journal of Management Information Systems*. **24**(3) 109-141

Lee, S. T., Z. Meng. 2005 Open source vs. proprietary software: competition and compatibility. Working paper, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=780804.

Lerner, J., J. Tirole. 2000. The simple economics of open source. Working paper, *National Bureau of Economic Research No. w 7600*.

Lerner, J., J. Tirole. 2001. The open source movement: key research questions. *European Economic Review*. **45**(4–6) 819–826.

Lerner, J., J. Tirole. 2002. Some simple economics of open source. *Journal of Industrial Economics*. **50**(2) 197–234.

Lilen, G., A. Rangaswamy. 1999. Diffusion models: managerial applications and software. *ISBM* Report. **7**.

Liu, L., P. P. Shenoy, C. Shenoy. 2003. A linear belief function approach to portfolio evaluation. *Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, 370–377.

Logsdon, J.M., J. K. Thompson, R. A. Reid. 1994. Software piracy: is it related to level of moral judgment? *Journal of Business Ethics*. **13** 849–857.

Marks, L. J., M. A. Kamins. 1988. The use of product sampling and advertising: effects of sequence of exposure and degree of advertising claim exaggeration on consumers' belief strength, belief confidence and attitudes. *Journal of Marketing Research*. **25** 266–281

Mustonen, M. 2003. Copyleft – the economics of Linux and other open source software. *Information Economic Policy*. **15**(1) 99–121.

Oliver, R.L. 1992. An investigation of the attribute basis of emotion and related affects in consumption: suggestions for a stage-specific satisfaction framework. *Advances in Consumer Research*. **19** 237–244

Peace, A. G., D. F. Galletta, J. Y. L. Thong. 2003. Software piracy in the workplace: a model and empirical test. *Journal of Management Information Systems*. **20**(1) 153–177.

Prasad, A., V. Mahajan. 2003. How many pirates should a software firm tolerate? — An analysis of piracy protection on the diffusion of software. *International Journal of Research in Marketing*. **20** 337–353.

Pynadath, D. V., S. C. Marsella. 2004. Fitting and compilation of multi-agent models through piecewise linear functions. *Proceeding of International Joint Conference on Autonomous Agents & Multi Agent Systems*, New York. **3** 1197–1204.

Rosenberg, D. K. 2000. *Open source: The unauthorized white papers*. IDG Book Worldwide Inc.

Sagers, G. W. 2004. The influence of network governance factors on success in open source software development projects. *Proceedings of the International Conference on Information Systems*. 427–438.

Sen, R. 2007. A strategic analysis of competition between open source and proprietary software. *Journal of Management Information Systems*. **24**(1) 233–257.

Shapiro, C. 1983. Optimal pricing of experience goods. *Bell Journal of Economics*. **14**(2) 497–507.

Shy, O., J. F. Thisse. 1999. A strategic approach to software protection. *Journal of Economics and Management Strategy*. **8** 163–190.

Sims, R. R., H.K. Cheng, H. Teegen. 1996. Toward a profile of student software pirates. *Journal of Business Ethics*. **15** 839–849.

Software and Information Industry Association (SIIA). 2000. SIIA report on global software piracy 2000.

Strassmann, A. P. 1998. Bogus payback: easy answers create long term problems. *Computerworld*.

Takayema,N.L. 1994. The welfare implications of unauthorized reproduction of intellectual property in the presence of demand network externalities. *Journal of Industrial Economics*. **42**(2) 155–166.

Tyson, L. D. A., E. F. Sherry. 1997. Statutory protection for databases: Economic and public policy issues. *Testimony before the Senate Committee on Judiciary*, October 23.

Venkatesh, D. L., S. Srinivasan, S. C. Kumbhakar. 1995. Software piracy: estimation of lost sales and the impact on software diffusion. *Journal of Marketing*. **59** 29–37.

Wheeler, D. A. 2004. Why open source software/free software (OSS/FS)? Look at the number! http://www.dwheeler.com/oss_fs_why.html.

Zwass, V. 1997. Editorial introduction. *Journal of Management Information Systems*. **13**(2) 3–6

BIOGRAPHICAL SKETCH

Yipeng Liu the only child of the family was born in Beijing, China. He grew up mostly in Beijing and graduated from Beijing Foreign Language High School in 1999 with a focus in English. He earned his B.E. in computer engineering and graduated with honors from the University of Science and Technology at Beijing, China in 2003. In the same year, Yipeng started his master's program in computer science at Southern Illinois University and worked on a thesis option track in the field of XML database query optimization with full support via graduate assistantship.

Upon graduating in August 2005 with his M.S. in computer science, Yipeng entered the Ph.D. program in information systems at the University of Florida and started working as a research and teaching assistant. His research involved two major fields: economic modeling of information systems and quantum computing algorithms for global optimization. Theses novel research ideas expanded results from computer science, economics, operation's research and statistics. Applications of this research could be found majorly in software industry in support decision making process for software managers. Yipeng presented his research at three international conferences, WITS 2007, AMCIS 2008, INFORMS 2008, and at the annual DIS research workshop held in February 2009 at the University of Florida. He was also invited to join the Dissertation Consortium at AMCIS 2008.

Yipeng graduated and received his Ph.D. from the University of Florida in the summer of 2009 and started an academic career in the Operations and Information Management Department at Kania School of Business at the University of Scranton.