The Effect of Compatibility on Software Supply and Hardware Demand: Forward and Backward Compatibility

Jae Nahm*

Hong Kong Univ. of Science and Technology

This paper looks at a hardware firm's decision to make its products forward and backward compatible in a model with heterogeneous consumer preferences. First, looking at a competitive hardware market, we show that backward compatibility always increases the valuation of hardware by the marginal consumer (who has the lowest valuation among purchasers of the new hardware), but backward compatibility can reduce the valuation of higher consumer types. Next, we look at a monopoly case. The effect of backward compatibility on the profit of the monopolist depends on how consumers substitute between old and new software programs and how sensitive the number of programs is to consumers' expenditures. The final part of the paper considers the case where the hardware firm has an option to employ an 'open licensing contract' on the software market, as is common in the video game industry. In this case, the firm uses the combination of the hardware price and a licensing fee to price discriminate between different consumer types. We study how compatibility decisions affect the price-discrimination scheme.

* Department of Economics, Hong Kong Univ. of Science and Technology. E-mail: jnahm@ust.hk, Tel: 852-2358-7626. I am grateful to Adam Brandenburger, Ricard Caves, Ethan Kaplan, Sonku Kim, Eric Maskin, Ariel Pakes, Michael Schwarz, Tomas Sjostrom, Catherine Wolfram, and to the participants of the Harvard Industrial Organization seminar and those of the Harvard Theory seminar. I thank Drew Fudenberg for guidance and support. All errors are my own.

1 Introduction

This paper builds a vertically-differentiated products model with heterogeneous consumer preferences and analyzes the effects of compatibility on software supply and hardware demand under three different market structures: a competitive hardware market; a monopoly hardware; and a market where a hardware monopolist has property rights on the software side.[1] This article clarifies the strategic nature of forward and backward compatibility decisions and shows that the effects of compatibility decision critically depend on a distribution of consumer preferences.

When an entity introduces a new system consisting of hardware and software, it makes a decision about the architecture of the new system. The architecture determines whether the new hardware technology can facilitate old-generation software (backward compatible); whether a new-generation software program can be used with the old-generation hardware technology (forward compatible).[2] Under forward compatibility, the old hardware can use new-generation software programs; however, it cannot utilize the software's full range of functions.[3] Under backward compatibility, new hardware facilitates old-generation software programs, but the combination of old-generation software programs and new hardware results in the same quality that the old hardware provides.[4] Independent software producers then decide whether to produce their programs for the new or old hardware plat-

[1]The hardware/software industry denotes any industry that deals with both a physical machine (hardware) and complementary programming (software). This includes, for example, the television industry, which has the actual TV as its hardware and TV programming as its software. Other examples include computers and computer software, Nintendo and Nintendo games, compact disc players and compact discs, and many other such markets.

[2]When two technologies are both forward and backward compatible, they are two-way compatible. When neither forward nor backward compatibility is supported, they are non-compatible.

[3]For instance, B/W TV receives color TV signals at B/W quality.

[4]For example, MS-Windows runs MS-DOS-based programs at the quality of MS-DOS

forms. Figure 1.1 illustrates the relationship between hardware and software under backward and forward compatibility.

Most previous studies have explored a firm's choice between two-way compatibility and non-compatibility (Farrel and Saloner (1992) and Choi (1996)). Also, they assume that the benefits from a technology increase monotonically with the size of consumer groups owning compatible technologies. If we infer the strategic implication of one-way compatibility from the assumption, it is always weakly better for a firm with a new technology to make its technology backward compatible, but not forward compatible with an existing technology. For instance, Farrell and Saloner (1992) argue that it is always desirable for a firm to offer a one-way converter with its product: to do so enhances the value of the firm's product without also enhancing its rivals'. However, these previous studies do not consider a software provider's incentive to invest in new-generation software. This paper shows that the implications of compatibility decisions can be quite different in the consumer-hardware-software framework. To motivate our model to readers, we offer two examples, one covering the case of forward compatibility and the other covering backward compatibility.

In 1950, the FCC voted to adopt CBS's color system as an industry standard.[5] This color system was, at the time, only backward compatible. That is, whereas color TV could receive black-and-white (B/W) signals, B/W TV could not receive color TV signals. According to previous studies, the optimal choice for color TV adoption is the backward- compatible color TV system. However, since a majority of household had B/W TV sets, broadcasting companies were not interested in airing major programs in color. And, because of the limited number of programs in color, consumers did not have an incentive to upgrade to color TV. Finally, in 1953, the FCC reversed its decision and adopted two-way compatible technology. That is, both B/W and color TV customers could receive color TV signals. After this change, broadcasting companies could send their color programs to both households

[5]See Farrell and Shapiro (1992)

with B/W and color TV sets. Therefore, they had much better incentives to develop color TV programming. Color TV programming increased rapidly, creating a larger consumer demand for color TV. This color TV example shows that supporting forward compatibility can actually be beneficial for the adoption of a new technology, contrary to the prediction of the existing models.

As an example of a product for which backward compatibility was an issue, consider DVD and DIVX.[6] DVD and DIVX are competing to capture the market for the next generation of digital formatting. A DIVX player not only plays all DVD discs, but it also plays special DIVX discs that cannot be run on DVD players. That is, DIVX is backward but not forward compatible with DVD. The DIVX platform seems to be a more attractive choice than the DVD platform. However, software companies can sell their DVD programs to consumers who own either a DVD or a DVIX player. Also, since the underlying differences between DVD and DIVX are relatively small, no large group of consumers has been willing to pay more for the extra feature available through DIVX. The prevalence of backward compatibility and the small number of consumers willing to pay for the extra feature of DIVX have convinced software companies to continue producing primarily for DVD players. In fact, as of May 31, 1999, there were 3,317 titles available in DVD format, but only 471 titles available in DIVX. The launch of DIVX has been unsuccessful so far, and DVD has become the market standard for digital formatting. This example shows that backward compatibility can actually 'backfire' the adoption of DIVX.

These two examples help clarify the strategic nature of forward and backward compatibility decisions. However, we cannot generalize the lessons from these two examples to all other cases. The effects of compatibility decisions critically depend on a distribution of consumer preferences. For instance, suppose that a majority of consumers highly differentiated the quality difference between color and B/W (corresponding to many "high types" in our

[6]See Dranove and Gandal (1999).

model) and bought color TV sets even with a small number of color TV programs. In this case, broadcasting companies would have switched away from B/W towards color programs even with the only-backward-compatible technology. Consumers with B/W TV would have had a very limited B/W TV programs. Even consumers who did not differentiate the quality difference between B/W and color would have been enforced to upgrade to color TV. With many high-type consumers, only-backward-compatible technology would have been the optimal choice for color TV adoption. Therefore, the optimal compatibility decision for a new technology adoption critically depends on the distribution of consumer preferences.

We need a more formal approach to evaluate these effects of compatibility on a firm's profits and new technology adoption. This paper builds a vertically-differentiated products model with heterogeneous consumer preferences. In our model, consumers of heterogeneous preferences generate and receive different sizes of network effects.[7] In this setting, we analyze how consumers and software companies react to these compatibility choices.

The central logic of this paper is illustrated in Figure 1.2. In a free-entry equilibrium, in which software companies just recover the fixed development cost, the variety and prices of software programs are endogenously determined by consumers' spending on software, and there exist positive externalities among consumers.[8] The compatibility decision determines a consumer's choice set at the software market.[9] When the choice set changes, a consumer changes her consumption pattern, which affects other consumers through software prices and variety. That is, compatibility choice has two effects on

[7]Most works on network externalities assume that all consumers within the same network generate and receive the same size of network effects.

[8]The price of a software program must be high enough for a limited consumer base. As the consumer base gets larger, software programs can recover the fixed development cost with a lower price. Therefore, as more consumers buy a hardware technology, more software programs for it can be supplied at lower prices.

[9]For instance, consumers can use only software programs of the same generation as their hardware under non-compatibility, while they can use software programs of both generations under two-way compatibility.

consumer surplus from hardware. First, it has the direct effect of changing a consumer's software choice set. Second, it has the indirect effect of changing the software prices and variety. The combined effects influence hardware demand.

The first part of this paper studies the case of a competitive hardware market. The main results of this section are, first, that backward compatibility always increases the marginal type's[10] valuation of the new hardware, but can reduce discriminating consumers' (high types') valuation. Therefore, implementing backward compatibility invokes a welfare trade-off even among consumers with the same hardware platform. We also show that forward compatibility can actually encourage the adoption of new technology, as in the case of color TV.

In the second part of the paper, we consider the case of a monopoly producer of new hardware. The price of a new hardware platform is the marginal types' willingness to pay for the additional benefits of new hardware over old hardware. Since backward compatibility increases the marginal type's valuation of new hardware, it has a positive effect on the monopoly supplier. However, it can increase the benefits of old hardware by increasing the number of old software programs. Therefore, backward compatibility can actually 'backfire' new technology. The effect of backward compatibility on the firm's profit depends on how backward compatibility affects the old-generation software market.

The third part of the paper explores how our analysis changes when the hardware company has property rights on both the hardware and software sides and has an option to employ a licensing contract on the software market, as is common in the video game industry. The firm can use the combination of the hardware price and a licensing fee to price discriminate between different consumer types. In this case, backward compatibility also involves a trade-off between profits from the hardware market and from the software market.

[10]The "marginal type" is the type who is indifferent between buying and not buying products.

Depending on parameters, this firm optimally chooses non-compatibility and tries to make old-generation software programs obsolete.

The paper is organized as follows: Section 1.1 reviews previous studies. Section 2 presents the formal model and imposes several restrictions. Sections 3, 4 and 5 study a competitive hardware market, hardware monopoly and a market where a hardware monopolist has property rights on the software side, respectively. Concluding remarks follow these analyses.

1.1 Literature

This paper is related to many lines of research. There is large literature on network effects.[11] However, this paper shows quite different implications of one-way compatibility from the existing literature. For instance, David and Bunn (1988) argue that in the battle between rival technologies, the adoption process and its formation of de facto standard may be decisively tipped by the development of a "one-way" converter that allows one of the technologies to obtain some of the network externalities accruing from the installed base of the other, but not vice versa. However, this paper shows that backward compatibility can actually backfire the adoption of a new technology and forward compatibility can help the adoption process.

There is 'Mix and Match' literature (Economides (1989) and Matutes and Regibeau (1988)). In their models, compatibility choice does not change the supply of complementary products. In our paper, compatibility choice, affecting the profitability of software providers, changes the supply of complementary software programs. Therefore, our model makes different predictions about the effects of compatibility on firms' profits.

Chou and Shy (1990) and Church and Gandal (1992) show how a network effect can arise with a CES utility function and free entry condition on the software market. They maintain the assumption that software pro-

[11] For a more comprehensive survey, see Katz and Shapiro (1994) and Benson and Farrell (1994). See, Choi (1994, 1996,1997) Economides (1989), Farrell and Saloner (1985, 1986), Katz and Shapiro (1985, 1986), among others.

grams are not compatible between the two hardware technologies though. Also, they implicitly assume that both hardware technologies are capable of performing the same tasks and that all consumers value software programs equally. Our paper builds a vertically-differentiated-product model in which heterogeneous consumers get different utility from the same network size and generate different network effects on other consumers.

2 A Simple Model

This section describes a model that includes hardware, software and consumers. The consumption of software programs provides no benefit without hardware and vice versa.

Hardware

There are two hardware technologies, old-generation and new-generation, denoted by hardware O and hardware N. The unit production costs for hardware O and N are H_O and H_N, respectively.

Software

Software programs written for hardware O and N are called old-generation software programs and new-generation software programs, respectively. Developing software programs incurs fixed costs, F_O and F_N respectively. Software programs are assumed to have zero marginal production cost.[12] At the beginning of the game, v_o old-generation software programs have already been developed and been available. Hardware N is introduced with a limited number of new-generation software programs, $\underline{v}$.

There is no entry barrier in the software industry. Therefore, the number of software companies in the market is endogenously determined to make the profits from developing a software program equal to the fixed cost of development.

[12] Our results will not change qualitatively even with a positive marginal production cost.

Even though consumers can consume multiple software programs, they generally consume only one unit of each program since additional copies of a software program do not provide consumers with higher utility.

The combination of hardware O and old-generation software programs yields low quality, while the combination of hardware N and new-generation software programs yields high quality. Whether a hardware technology works with software programs of a different generation depends on compatibility decisions.

Compatibility

The entity that introduces the new system decides about the architecture of the new system. The architecture determines whether the new hardware technology can facilitate old-generation software; whether a new-generation software program can be used with the old-generation hardware technology.

(1) Non-compatibility (NC): Under non-compatibility, a hardware technology can facilitate only software programs of the same generation. The combination of hardware and software programs from different generations produces no benefit.

(2) Backward compatibility (BC): Under backward compatibility, hardware N can facilitate both old-generation and new-generation software programs. However, the combination of hardware N and old-generation software programs still provides low quality.

(3) Forward compatibility (FC): Under forward compatibility, hardware O can facilitate both new-generation and old-generation software programs. However, the combination of hardware O and new-generation software programs still provides low quality.

(4) Two-way compatibility: When both backward and forward compatibility are supported, the new technology is two-way compatible with the old technology.

Consumers

Consumers' preferences are denoted by a quasi-linear utility function,

$U(x, y; t) + M$, $t = h, m$, or l, where M is the amount of money spent on outside goods, and $U(x, y; t)$ represents the utility level that a type t receives from consuming x old-generation software programs and y new-generation software programs. There are three types of consumers: types h (high), m (middle) and l (low). All types get the same utility from consuming old-generation software programs, but they differ in their willingness to pay for new-generation software programs.

Assumption 1: $U_x(x, y; l) = U_x(x, y; m) = U_x(x, y; h) = U_y(x, y; l)$
$U_y(x, y; l) < U_y(x, y; m) < U_y(x, y; h)$.

Assumption 1 implies that all three types have the same marginal utility of old-generation software programs; low-type consumers are indifferent between old-generation and new-generation software programs; high-type consumers highly differentiate between them; and middle-type consumers are between low-type consumers and high-type consumers in this regard. There are N_h high-, N_m middle-, and N_l low-type consumers.

Consumers maximize their utility by choosing optimal x and y at the software market with two constraints.[13] One is a budget constraint, $M = B - \sum_{i=1}^{x} p_{oi} - \sum_{i=1}^{y} p_{ni}$, where B is a total budget and p_{oi} and p_{ni} are the prices of old and new-generation software programs, respectively. Software products are labeled so that lower index "i" means lower price. $\sum_{i=1}^{x} p_{oi}$ and $\sum_{i=1}^{y} p_{ni}$ denote the total expenditures on old-generation and new-generation software, respectively.

The other constraint is that consumers' choice sets are determined by their hardware platforms and compatibility. For instance, under non-compatibility, consumers owning hardware O will maximize $U(x, 0; t) + M$ by choosing only old-generation software programs.[14]

[13] x and y must be integers. However, we treat x and y as continuous variables since there is no integer problem in this paper.

[14] There is another constraint that the number of software programs consumers want to

Since different software programs are available for a different hardware technology, each hardware generates a different consumer surplus. Consumers choose hardware N when the additional consumer surplus of hardware N over hardware O is larger than the price difference between hardware N and O.

The timing of moves

The timing of moves is illustrated in Figure 2.1. At the beginning of the game, the entity that introduces the new system makes its compatibility decision. Consumers buy their hardware technologies. Then, software companies enter the software market by developing software programs. Consumers make their optimal purchase of software programs.

Parameter restrictions

The number of old-generation software programs that have already been developed is v_o. We assume that v_o is large enough that the profits from developing additional old-generation software program is less than the development cost. Therefore, the number of old-generation software programs is fixed at v_o. Assumption 2 is a sufficient condition for this.[15] In section 4.2, we will comment on how implications of our results change when Assumption 2 is relaxed.

Assumption 2 $U_x(v_o, 0; l) < \frac{F_O}{N_l + N_m + N_h}$

As will be shown later, Assumption 2 implies that the price of an old-generation software program is uniformly $U_x(v_o, 0; l)$. For convenience, we denote $U_x(v_o, 0; l)$ by $\bar{p}_x$. At this price, each consumer owning hardware O buys v_o old-generation software programs . The consumer surplus from hardware O becomes CS_O, where $CS_O = U(v_o, 0; t) - \bar{p}_x v_o$.

buy must be less than or equal to the number of programs actually available. However, the constraint is not binding because as we shall see, we posit that the prices of software will be set such that it becomes irrational for consumers to demand in excess of what the market will provide.

[15] Lemma One in section 3.1 will show this.

Hardware N is introduced with $\underline{v}$ new-generation software programs. Therefore, the minimum value of hardware N is $\underline{CS}_N(t)$, where $\underline{CS}_N(t) = U(0, \underline{v}; t) - U_y(0, \underline{v}; h)\underline{v}$.[16] As more consumers buy hardware N, more software programs will be supplied at lower prices.[17] Therefore, the consumer surplus from hardware N will be determined by how many consumers buy hardware N. For instance, $CS_N(h, m, l; t)$ denotes consumer surplus of type t from hardware N when all three types choose hardware N. $CS_N(h; t)$ denotes consumer surplus of type t when only high-type consumers buy hardware N.

We assume that all high-type consumers optimally buy hardware N and all low-type consumers optimally buy hardware O when these hardware platforms are supplied at marginal production cost, H_O and H_N.

Assumption 3.1 $CS_O - H_O > CS_N(h, m, l; l) - H_N$
Assumption 3.2 $CS_O - H_O < \underline{CS}_N(h) - H_N$

Assumption 3.1 is a sufficient condition for low-type consumers to buy hardware O. The left-hand side, the net consumer surplus from hardware O, is larger than the right-hand side, the maximum net consumer surplus from hardware N for low-type consumers.[18] Similarly, Assumption 3.2 is a sufficient condition for high-type consumers to buy hardware N. The left-hand side, the net consumer surplus from hardware O, is less than the right-hand side, the minimum net consumer surplus from hardware N for the high-type consumer.

With Assumption 3.1 and Assumption 3.2, this paper focuses on the hardware decision of the middle-type consumers. As will be shown, the middle type's hardware decision critically depends on the compatibility decision.

[16]The minimum number of new-generation software programs is $\underline{v}$. The potentially highest price for the new-generation program is $u_y(0, \underline{v}; h)$. Therefore, the minimum benefit of owning hardware N is $\underline{CS}_N(t) = U(0, \underline{v}; t) - u_y(0, \underline{v}; h)\underline{v}$.

[17]Lemma One in section 3.1 will show this result.

[18]The highest consumer surplus from hardware N is achieved when all consumers buy hardware N.

Equilibrium Selection

Since the variety and prices of new-generation software programs depend on how many consumers buy hardware N, there exists a coordination problem among consumers. According to expectations, we can have multiple equilibria. Therefore, without an equilibrium selection, we cannot analyze companies' strategic choices.[19] Similar to the approach taken by Katz and Shapiro(1986), this paper assumes that decisions are made as if middle-type consumers can coordinate their choices.

Assumption 4 Consumers of middle type can coordinate to get their best payoff.

3 A competitive hardware market.

This section studies the case in which these hardware technologies are non-proprietary and are supplied in a competitive market at marginal cost, H_O and H_N. We will solve for consumers' optimal consumption of software programs and hardware technology under non-compatibility, backward compatibility, and forward compatibility.

3.1 Non-compatibility

Under non-compatibility, a hardware technology facilitates only software programs of its generation. Therefore, consumers owning hardware N will buy only new-generation software programs. They maximize $U(0, y; t) + M$, (s.t) $M = B - \sum_1^y p_{ni}$ by choosing y. $y^t(p_y)$ denotes the type t's optimal consumption of new-generation software programs when they sell uniformly at price p_y.[20] Consumers owning hardware O maximize $U(x, 0; t) + M$, (s.t) $M =$

[19] It would be of interest how consumers' expectations are formed, but we set aside the issue.

[20] $y^t(p_y)$ is $\arg\max U^t(0, y; t) - p_y y$.

$B - \sum_1^x p_{oi}$ by choosing x. When each old-generation software sells uniformly at price p_x, $x^*(p_x)$ denotes the optimal consumption of these programs.[21]

The prices of software programs are endogenously decided by consumers' consumption patterns in free-entry equilibrium. As the size of the consumer group owning hardware N increases, a software company can sell more units of a software program and recover its fixed cost with a lower price. At the free-entry equilibrium, in which software companies just recover the fixed development cost, more software programs for it can be supplied at lower prices with a larger consumer group.

Let us solve how these prices are determined in a general subgame where k consumers have hardware O and b middle-type consumers and d high-type consumers have hardware N.[22]

Suppose that there are α old-generation software programs. In this case, all of these α programs sell k units, each at price $U_x(\alpha, 0)$. In the old-generation software market, v_o old-generation software programs have already been developed. By Assumption 2, the marginal utility of the v_oth old-generation software program is lower than $\frac{F_O}{N_l+N_m+N_H}$. Therefore, a company cannot recover the fixed development cost even with selling its program to all consumers owning hardware O. There will be no additional old-generation software development. All these v_o software programs sell uniformly at price $\bar{p}_x = U_x(v_o, 0; l)$.

Figure 3.1 shows the relationship between software prices and variety in the new-generation software market. A software company cannot recover the fixed cost F_N with a lower price than $\frac{F_N}{b+d}$ since a new-generation program can sell at most $b+d$ units. Therefore, the lowest price for a new-generation program is $\frac{F_N}{b+d}$. At this price, consumer type m wants $y^m(\frac{F_N}{b+d})$ programs. If $y^m(\frac{F_N}{b+d})$ programs are not supplied at this price, then a new firm can enter, sell to all $b+d$ consumers, and make a profit, contradicting free-entry equilibrium. Therefore, $y^m(\frac{F_N}{b+d})$ programs are supplied at price $\frac{F_N}{b+d}$.

[21] $x^*(p_x)$ is $\arg\max U(x, 0; t) - p_x x$.

[22] Under Assumption 3, all low-type consumers optimally buy hardware O. Therefore, we exclude the case where a low-type consumer buys hardware N.

Since the high type gets higher utility than the middle type, high-type consumers want to buy more programs. Since an additional program can be sold at most d units, the price of an additional program must be at least $\frac{F_N}{d}$ to recover the fixed cost. If high type's willingness to pay for an additional program is higher than this price, high type buys additional $(y^h(\frac{F_N}{d}) - y^m(\frac{F_N}{b+d}))$ programs, and by free entry they must all be supplied.

Lemma 1 *Suppose that b middle-type consumers and d high-type consumers purchase hardware N. At the free-entry equilibrium, every consumer owning hardware O buys v_o old generation software programs, each at price $\bar{p}_x$, where $\bar{p}_x = U_x(v_o, 0)$. Every consumer owning hardware N buy $y^m(\frac{F_N}{b+d})$ new-generation software programs, each at price $\frac{F_N}{b+d}$. When $y^h(\frac{F_N}{d})$ is higher than $y^m(\frac{F_N}{b+d})$, every high-type consumer purchases additional $(y^h(\frac{F_N}{d}) - y^m(\frac{F_N}{b+d}))$ programs, each at price $\frac{F_N}{d}$.*

Proof. See Appendix (a) ■

As a result, some new-generation software programs sell high volumes at low price, and some new-generation software programs sell low volumes at high price.[23] All make zero profit in the new-generation software market. The number of software programs that are supplied at a lower price $(\frac{F_N}{b+d})$ is decided by the number of programs that a middle type buys. The more middle-type consumers spend on new-generation software, the larger their externality on high-type consumers is.

Let us study middle type's hardware decision. Since each consumer owning hardware O can buy old-generation software programs each at $\bar{p}_x$, the consumer surplus from hardware O is CS_O, where CS_O is $U(v_o, 0; t) - \bar{p}_x v_o$. When all consumers of middle type choose hardware N, by lemma one, every middle-type consumer buys $y^m(\frac{F_N}{N_m+N_h})$ new-generation software programs each at price $\frac{F_N}{N_m+N_h}$. The consumer surplus of middle type from

[23] In this case, $y^m(\frac{F_N}{b+d})$ new-generation programs sell $b+d$ units, each at price $\frac{F_N}{b+d}$. $(y^h(\frac{F_N}{d}) - y^m(\frac{F_N}{b+d}))$ new-generation programs sell d units, each at price $\frac{F_N}{d}$.

hardware N is $CS_N(h, m; m)$, where $CS_N(h, m; m)$ is $U(0, y^m(\frac{F_N}{N_m+N_h}); m) - (\frac{F_N}{N_m+N_h})y^m(\frac{F_N}{N_m+N_h})$.

$CS_O - H_O < CS_N(h, m; m) - H_N - --$ **(Equation 1)**

Equation 1 shows the condition under which middle-type consumers choose hardware N. We will study how backward compatibility changes this condition.

3.2 Backward Compatibility

This section studies the effects of backward compatibility on consumer surplus from hardware N. Under backward compatibility, consumers owning hardware N can use both old and new-generation software programs. They maximize $U(x, y; t) + M$, (s.t) $M = B - \sum_1^x p_{oi} - \sum_1^y p_{ni}$ by choosing x and y. Since old-generation software programs and new-generation software programs are imperfect substitutes for each other, the marginal utility of new-generation software programs depends on how many old-generation software programs an individual has.

(BC1) $U_{xy}(x, y; t) < 0$.

(BC1) assumes that the cross derivatives of the utility function with respect to x and y are negative. That is, the more old-generation software programs an individual has, the lower the marginal utility of new-generation software program is.

Even if hardware N can technically facilitate software programs of both generations, we can have corner solutions where individuals buy only new-generation software. (BC2) and (BC3), comparing the MRS[24] between new and old-generation software programs with the ratio of two prices, rule out the corner solutions.

[24]Marginal rate of substitution

(BC2) $\frac{U_y(0,y;\, m)}{U_x(0,y;\, m)} < \frac{F_N}{N_h+N_m}\frac{1}{\bar{p}_x}$.

(BC3) $\frac{U_y(0,y\,;h)}{U_x(0,y\,;h)} < \frac{F_N}{N_h}\frac{1}{\bar{p}_x}$.

(BC2) and (BC3) are sufficient for middle-type consumers and high-type consumers owning hardware N to buy some old-generation software programs. Without (BC2) and (BC3), backward compatibility is meaningless since consumers owning hardware N will buy no old-generation software program.

Backward compatibility changes consumers' choice sets of software programs and thereby consumers' expenditures on software programs. Therefore, backward compatibility brings in a different set of software prices and variety. These changes in the software prices and variety have different effects on consumers with heterogeneous preferences.

Let us study how backward compatibility affects the hardware decisions of middle-type consumers. In the case where middle type consumers buy hardware N, a consumer of middle type can buy new-generation software programs each at $\frac{F_N}{N_m+N_h}$. Middle-type consumers maximize $Max\ U(x,y;m)$ (s.t) $B = \bar{p}_x x + (\frac{F_N}{N_m+N_h})y$. Their optimal number of new-generation software program under backward compatibility is represented by y^m_{BC}.[25] Therefore, the number of new-generation software programs that are sold at $\frac{F_N}{N_m+N_h}$ is y^m_{BC}. Under backward compatibility, they can buy their optimal number of new-generation software programs at the same price as under the non-compatibility case**.** Also**,** they can enjoy old-generation software. Therefore, middle type's consumer surplus from hardware N is higher under backward compatibility than under non-compatibility ($CS^{BC}_N(h,m;m) > CS_N(h,m;m)$). Equation 2 shows the condition in which middle-type consumers buy hardware N. Equation 2 is weaker than Equation 1. That is, middle-type consumers are more likely to adopt hardware N under backward compatibility.

[25]The optimal consumption problem of a middle type is $max\ U(x,y;m) - \bar{p}_x x - (\frac{F_N}{N_m+N_h})y$. The optimal x^m_{BC} and y^m_{BC} are such that $mu^m_x(x^m_{BC}, y^m_{BC}) = \bar{p}_x$ and $mu^m_y(x^m_{BC}, y^m_{BC}) = \frac{F_N}{N_m+N_h}$. The middle type's consumer surplus from hardware N ($CS^{BC}_N(h,m;m)$) becomes $U(x^m_{BC}, y^m_{BC}; m) - \bar{p}_x x^m_{BC} - (\frac{F_N}{N_m+N_h})y^m_{BC}$.

$CS_O - H_O < CS_N^{BC}(h, m; m) - H_N - --$ **(Equation 2)**

Let us study how backward compatibility affects the high type's consumer surplus from hardware N. Let us look at the following three scenarios. In the first scenario, middle-type consumers do not buy hardware N no matter what the compatibility decision is. Under backward compatibility, the high type buy new-generation software programs at the same price as under non-compatibility and can use old-generation software. Therefore, high type's consumer surplus from hardware N increases with backward compatibility.

Figure 3.2A shows the second scenario where middle-type consumers buy hardware N only under backward compatibility. Under non-compatibility where only high-type consumers choose hardware N, each new-generation software program sells at price $\frac{F_N}{N_h}$. Under backward compatibility, y_{BC}^m new-generation software programs are sold, each at price $\frac{F_N}{N_m+N_h}$. Therefore, high-type consumers buy these y_{BC}^m programs at lower price under backward compatibility than under non compatibility. Backward compatibility increases the consumer base for new-generation software programs and has a positive externality on high-type consumers.

Figure 3.2B shows the third scenario where middle-type consumers choose hardware N under both non-compatibility and backward compatibility. Here, backward compatibility does not increase the size of the consumer base for new generation software programs. Middle-type consumers buys fewer new-generation software programs under backward compatibility than under non-compatibility. Therefore, the number of programs sold at a lower price declines, which has a negative externality on high-type consumers. The following example illustrates the case where backward compatibility indeed decreases high type's consumer surplus from hardware N.

Example 1 $U(x, y; m) = \log(x+y)+\log(y), U(x, y; h) = \log(x+y)+2\log(y)$, $\bar{p}_x = 0.05$, $\frac{F_N}{N_m+N_h} = 0.2$, *and* $\frac{F_N}{N_h} = 0.5$. *Under non-compatibility, 10 new-generation software programs are sold, each at price* 0.2. *However, under*

backward compatibility, only $\frac{20}{3}$ *new-generation software programs are sold, each at price* 0.2. *The high-type consumer's surplus from hardware N under non-compatibility is* $3\log(10) - 2$, *which is higher than the consumer surplus under backward compatibility,* $\log(20) + 2\log(\frac{20}{3}) - 2$. *Thus backward compatibility can actually reduce high type's consumer surplus from hardware N.*

Backward compatibility does not improve every consumer's surplus from hardware N and involves a trade-off even among consumer groups with the same hardware.

Proposition 1 *With backward compatibility comes a new set of software prices and varieties, which have different effects on different consumer groups. Backward compatibility always increases the valuation of hardware by a marginal type, although it can actually reduce the valuation of higher types.*

3.3 Forward Compatibility and Two-Way Compatibility

Under forward compatibility, hardware O can facilitate both new and old-generation software programs. Software companies can sell new-generation software programs to both consumer groups, one with hardware N and the other one with hardware O. Consumers owning hardware O purchase their software programs based on price because those old-generation and new-generation software programs are perfect substitutes for them.

(FC1) $\frac{F_N}{N_h+N_m+N_l} < \bar{p}_x = U_x(v_o, 0)$

(FC1) implies that the fixed cost per capita of the new-generation software program is lower than $U_x(v_o, 0)$ (the marginal utility of the v_oth old-generation software program). If (FC1) does not hold, the (potentially) lowest price of a new-generation program is higher than that of old-generation program, so consumers owning hardware O will not buy new-generation software. That is, forward compatibility is meaningless in the case (F1) does not hold.

With (FC1), some new-generation programs can be supplied each at $\frac{F_N}{N_h+N_m+N_l}$, and by free entry they must all be supplied. Consumers owning hardware O want to buy v^F programs at $\frac{F_N}{N_h+N_m+N_l}$.[26] Therefore, $(v^F - v_o)$ new generation programs are supplied each at $\frac{F_N}{N_h+N_m+N_l}$.[27]

Lemma 2 *Under forward compatibility with (FC1), v_o old-generation programs sell each at $\frac{F_N}{N_h+N_m+N_l}$. Some of new-generation program sell each at $\frac{F_N}{N_h+N_m+N_l}$.*

$$CS_O^{FC} - H_O < CS_N^{FC}(h, m; m) - H_N - \text{-- } \textbf{(Equation 3)}$$

Equation 3 shows when middle type buys hardware N under forward compatibility. Consumers owning hardware O buy v_o old-generation software programs and some of new-generation software programs each at $\frac{F_N}{N_h+N_m+N_l}$. In the comparison with non-compatibility, the benefit of hardware O increases. Also, some of new-generation programs are supplied each at $\frac{F_N}{N_h+N_m+N_l}$, which increases the benefit of hardware N. Therefore, forward compatibility increases the benefits of hardware N as well as hardware O. The net effect of forward compatibility on the middle type's decision to buy hardware N depends on the relative size of these two effect. The following example shows the case in which forward compatibility increases the adoption of hardware N.

Example 2 *Suppose that* $U(x, y; m) = \log(x + y) + \log(y)$, $H_N - H_O = \log(\frac{36}{5}) - 0.6$, $\bar{p}_x = 0.5$ $\frac{F_N}{N_m+N_h} = 0.33$ *and* $\frac{F_N}{N_l+N_m+N_h} = 0.2$. *Then, the consumer surplus of type m from hardware technologies are follows,* $CS_O^{NC} = \log(2)-1$, $CS_N^{NC}(m, h; m) = 2\log(6)-2$, $CS_O^{FC} = \log(5)-1$, *and* $CS_N^{FC}(m, h; m) =$

[26] v^F is $arg\ max U(x, 0; t) - \frac{F_N}{N_h+N_m+N_l}x$. Please note that old and new-generation programs are perfect substitute each other for consumers owning hardware O.

[27] These new-generation programs sell k units to consumers owning hardware O and $(N_h+N_m+N_l-k)$ units to consumers owning hardware N when the number of consumers with hardware O is k.

$2\log(6) - 1.6$. *Type m chooses hardware O under non-compatibility but hardware N under forward compatibility. Therefore, forward compatibility increases the adoption of new technology.*

Proposition 2 *Forward compatibility not only protects consumer groups with hardware O, but can also increase the benefits from hardware N. Forward compatibility can, thus, increase the adoption of hardware N.*

This result suggests an explanation for the color TV case. When consumers were slow in upgrading to the color TV, broadcasters were reluctant to send their programs in color since the majority of households had B/W system. Consumers also did not have a reason to buy color TV with a limited number of color programs. Therefore, it was difficult to break a chicken-egg problem between software and households without supporting forward compatibility.

Remark 1 ***Two-way compatibility***

Previous studies analyze a firm's incentive to adopt two-way compatibility by assuming that beneficial network effects increase monotonically in the consumer group with compatible technologies. However, in the setting of consumer-hardware-software, the effects of two-way compatibility depend on the size of fixed costs and the distribution of consumer preferences. For instance, suppose that (FC1) does not hold under two-way compatibility. Consumers owning hardware O buy only old-generation software programs and consumers owning hardware N buy both old and new-generation software programs, as in the case of backward compatibility. Therefore, understanding one-way compatibility is essential in understanding two-way compatibility.

4 Hardware Market Monopoly

This section investigates the effects of the compatibility decision on the monopoly profitability. From now on, we will analyze the case where hardware N is supplied by a monopolist and hardware O and software are still

supplied by a competitive market. This section identifies the conditions under which backward compatibility increases the monopoly profits.

4.1 Non-compatibility

We have assumed that the low-type consumers will not buy hardware N even if it is supplied at the marginal unit production cost. Therefore, the firm does not consider lowering its price to entice low-type consumers to buy hardware N. The firm's optimal strategy is either selling hardware N to both high-type and middle-type consumers or selling hardware N to only high-type consumers.

The monopoly supplier of hardware technology N cannot price discriminate among consumers and set a uniform price for all consumers. Therefore, the marginal consumer's, rather than the average consumer's, valuation of the product decides the market price. When the high-type consumer is the marginal type, a high-type consumer is willing to pay $p_N(h)$ for hardware N, where $p_N(h)$ is $[CS_N(h;h) - CS_O + H_O]$. The firm's profit is $\Pi^{NC}(h) = [p_N(h) - H_N]N_h$. When the middle-type consumer is the marginal type, a middle-type consumer is willing to pay $p_N(m)$, where $p_N(m)$ is $[CS_N(m,h;h) - CS_O + H_O]$. In this case, the firm's profit is $\Pi^{NC}(m,h) = [p_N(m) - H_N][N_m + N_h]$.

4.2 Backward compatibility

This section studies how backward compatibility changes the marginal type's consumer surplus from hardware N and affects the firm's profitability. When the high type is the marginal type, the high-type consumer is willing to pay $p_N^{BC}(h)$ for hardware N, where $p_N^{BC}(h) = CS_N^{BC}(h;h) - CS_O + H_O$. Similarly, when the middle type is the marginal type, the middle type consumer is willing to pay $p_N^{BC}(m)$ for hardware N, where $p_N^{BC}(m)$ is $CS_N^{BC}(m,h;m) - CS_O + H_O$.

As section 3.2 has shown, no matter which type is the marginal type, the

marginal type gets a higher consumer surplus from hardware N under backward compatibility than under non-compatibility. Therefore, the monopoly profitability is higher under backward compatibility ($\Pi^{BC}(h) > \Pi(h)$ and $\Pi^{BC}(m,h) > \Pi(m,h)$).

However, critical to this result is Assumption 2 that enough varieties of old-generation have already been developed and the varieties do not change between non-compatibility and backward compatibility. When we relax Assumption 2, backward compatibility can actually increase the variety of old-generation software, which has a negative effect on the profitability of hardware N. That is, backward compatibility can "backfire" against hardware N. Therefore, the effect of backward compatibility on the firm's profits depends on how software companies react to backward compatibility on old-generation software market.

Proposition 3 *Monopoly profitability is higher under backward compatibility than under non-compatibility. However, when Assumption 2 is relaxed, backward compatibility can increase old-generation software variety, which has a negative effect on the monopoly profits. Therefore, the effect of backward compatibility on profits depends on how backward compatibility affects the old-generation software market.*

The case of DVD vs. DIVX is one example of the "backfire". Since the underlying differences between DVD and DIVX are relatively small, there has not been a large group of consumers willing to pay much extra in order to get the extra feature available through DIVX. In addition, software companies can sell their products in DVD format to both consumer groups with DVD and DIVX. As a result, software companies have primarily continued to produce for DVD only, and DVD has become the market standard.

5 The hardware company has controls on the software market

So far we have analyzed the relationship between monopoly hardware pricing and compatibility decision when the monopolist does not have property rights on the software market. This section studies the relationship between them when the monopolist has property rights on software as well as hardware.

Since it is difficult for the monopolist to produce all software programs for its hardware, the monopolist allows other companies to supply software. There are several strategies the monopolist can employ. In this paper, we look at cases where the monopolist uses an open licensing policy: Any software company can supply its software for hardware N as long as it pays the licensing fee set by the monopolist. (The licensing contract is 'open' in that sense.) The licensing fee allows the monopolist to make profits from the software market as well as the hardware market.

We assume that the monopolist sets its software licensing fee and its hardware price together at the beginning of the game.[28] After that, consumers choose their hardware technologies. Then software companies supply new-generation software on the terms set by the monopolist. This section shows that the firm can use a licensing fee as price discrimination device and increase its profit.

5.1 Non-Compatibility

On the software market, any software company can supply programs for hardware N by paying a variable fee (or royalty) f per unit. For instance, when a software company sells k units of one program, it pays kf to the monopolist.

With this licensing contract, the marginal unit production cost becomes

[28]Even though there is a time inconsistency problem in the firm's optimal licensing fee (the monopoly optimal licensing fee is different between before selling and after selling its hardware), we assume that the monopoly can make a commitment to its licensing fee.

f. By lemma one, when all middle-type and high-type consumers purchase hardware N, every consumer owning hardware N buys $y^m(\frac{F_N}{N_h+N_m}+f)$ new-generation software programs each at price $\frac{F_N}{N_h+N_m}+f$. When $y^h(\frac{F_N}{N_h}+f) > y^m(\frac{F_N}{N_h+N_m}+f)$, every high-type consumer purchases additional $(y^h(\frac{F_N}{N_h}+f) - y^m(\frac{F_N}{N_h+N_m}+f))$ programs each at price $\frac{F_N}{N_h}+f$. The prices of new-generation software increase and the number of software programs declines. Therefore, a licensing fee lowers consumer surplus from hardware N, which is internalized into a hardware price drop.

However, the licensing fee generates profits from the software market. Therefore, there is a trade-off between the profits from the hardware market and the profits from the software market in setting a licensing fee. Thus, the monopolist will charge a positive licensing fee only if the profits gain from the software market are larger than the profits loss in the hardware market.

The hardware price will be set equal to the marginal type's willingness-to-pay for hardware N. When the high type is the marginal consumer to buy hardware N, the loss of consumer surplus is always larger than the revenue generated by the licensing fee. Figure 5.1 shows this result clearly. The licensing fee reduces the consumer surplus from hardware N, which is measured by area (A). Part of this loss is captured by the firm as profits from the licensing fee, which is measured by area (B). The area of (B) is always less than the area of (A). The area of (C), which is (A)-(B), represents the dead-weight loss generated by the licensing fee. Therefore, the combined profits are maximized with a zero licensing fee ($f = 0$).

Let us analyze the case where middle type is the marginal type. Figure 5.2 shows the case. The licensing fee decreases the consumer surplus of middle type and high type consumers differently. However, since the hardware price is decided by a marginal type's valuation, the change in hardware price is entirely determined by the change in the middle type's consumer surplus. The price drop is measured by area (a)+(c). The profits generated by a middle-type consumer on the software side are measured by area (a) and are always less than the hardware price drop ((a)+(c)). Therefore, the licensing

fee decreases the total revenue from middle-type consumers. However, since the high-type consumers buy a different number of programs, the profits generated by high-type consumers on the software side are measured by area (a)+(b) and can be higher than the hardware price drop. Therefore, the monopolist can increase the total revenue from high-type consumers. If the increase in revenue from the high-type consumers is larger than the decrease in revenue from the middle-type consumers (ie. $N_h b$ is larger than $(N_h + N_m)c$), the combined profits increase with the licensing fee. The following example shows that the firm's profit increases with a positive licensing fee.

Example 3 $U(x, y; m) = log(x + y) + log(y) + 3$, $U(x, y; h) = log(x + y) + 5\log(y) + 3$, $F_N = N_H = N_M = 1$ ($\frac{F_N}{N_H} = 1, \frac{F_N}{N_H + N_M} = 0.5$). *Without licensing fee, middle type consumers buy 4 new generation software products and get 2log(4)+1 consumer surplus from the new system. Therefore, the firm can set the hardware price at 2log(4)+1, and its profit becomes 4log(4)+2. However, with setting* $f = 0.25$, *middle type consumers buy* $\frac{8}{3}$ *new generation software products and get* $2log(\frac{8}{3}) + 1$ *consumer surplus. However, the firm can generate profits with licensing fee at the software market,* $\frac{2}{3}$ *from a middle type consumer and* $\frac{6}{5}$ *from a high type consumer. These extra revenue from the software market are larger than the hardware price drop. Therefore, the firm can increase its profits with a positive licensing fee.*

Proposition 4 *When high type is the marginal type, the firm's optimal licensing fee is zero, which is equivalent to giving up its property right on the software market. When middle type is the marginal type, the firm can price discriminate among consumers more effectively and increase profits with the combination of hardware price and licensing fee.*

A positive licensing fee increases prices of software, which has different effects on middle-type and high-type consumers. The firm can price discriminate between consumers and can increase its profitability with a positive licensing fee.

Remark 2 *(Organizational Issue)*

When the firm has control rights on both hardware and software markets, one organizational issue is how the firm should control the software market. The firm has two choices: One is to set a zero licensing fee and extract the maximized consumer surplus with a hardware price. The other is to price discriminate between consumers with a combination of a hardware price and a licensing fee. The firm's optimal choice is determined by the relative size of $N_h b$ and $(N_h + N_m)c$ in figure 5.2. As the size of middle type consumers gets larger, the optimal licensing fee becomes zero. However, when N_h gets larger, the firm can increase its profits with a positive licensing fee. Also, the size of b is related to the preference difference between the middle and the high type. When their preferences are sufficiently homogenous, the size of b gets closer to zero, and the optimal licensing fee becomes zero. That is, the firm's optimal organization form depends on the distribution of consumer preferences.

5.2 Backward compatibility

Nintendo decided against backward compatibility between their new 16-bit system and their existing 8-bit games. Some analysts criticized the incompatibility decision. Backward compatibility would have been a strong selling point for the 16-bit system because of their massive stock of Nintendo 8-bit software.[29] This section shows that the monopolist optimally can choose non-compatibility when the firm can also make profits from the software market with a licensing fee.

Since backward compatibility increases the marginal type's valuation of hardware N, it increases the firm's profits from the hardware market. However, since consumers owning hardware N spend less on new-generation software under backward compatibility, it decreases the demand for new-generation software and the monopolist's profits from the software market.

[29] See Brandenburger and Nalebuff (1996).

Therefore, backward compatibility involves a trade-off between profits from the hardware side and the software side.

When the firm's optimal licensing fee is zero under non-compatibility, its profits come from only hardware sales. Therefore, backward compatibility is the optimal compatibility choice for the firm.

However, when the optimal licensing fee is positive under non-compatibility, backward compatibility involves the trade-off. The optimal compatibility choice is determined by the relative sizes of the profit gain from the hardware market and the profit loss from the software market.

The relative size depends on how consumers substitute between old-generation and new-generation software. For instance, let us consider an extreme case where the marginal utility of new-generation software does not depend on old-generation software programs. In this case, consumers do not reduce their expenditure on new-generation software. Therefore, backward compatibility increases the profits from the hardware market without reducing the profits from the software market. Backward compatibility is the optimal compatibility choice for the firm. However, when the extent of substitution between old- and new-generation software is sufficiently strong, backward compatibility decreases the profits from the software market more than the profit gained from the hardware market. In this case, the firm optimally chooses non-compatibility and tries to make old-generation software obsolete, even though the monopolist can technologically support backward compatibility in hardware N. Therefore, the optimal compatibility choice depends on how homogeneous consumers are and how they substitute between old and new-generation software.

Proposition 5 *When the monopolist can make profits from the software market with a licensing fee, backward compatibility involves a trade-off between the profits from the hardware market and the profits from the software market. The relative size determines the firm's optimal compatibility. When the extent of substitution between old- and new-generation software is sufficiently strong, the firm is more likely to choose non-compatibility.*

Example 4 $U(x,y;m) = log(x+y) + log(y) + 3$, $U(x,y;h) = log(x+y) + 5\log(y) + 3$, $F_N = N_H = N_M = 1$ ($\frac{F_N}{N_H} = 1, \frac{F_N}{N_H+N_M} = 0.5$),*and* $\bar{p}_x = 0.4$. *Under non-compatibility, the optimal licensing fee is 0.35, and the profits are 7.80. Under backward compatibility, the optimal licensing fee is 0.3, and the profits are 7.79, which is lower than the profits under non-compatibility. Therefore, non-compatibility can be an optimal compatibility choice for the firm with control rights on the software market.*

Remark 3 *Planned obsolescence.*

With making its hardware non-compatible (rather than backward compatible) with the previous standard, the firm reduces consumers' spending on old-generation software and make obsolete the old-generation hardware. This topic is related to planned obsolescence, including Choi (1994) and Waldman (1993). Choi (1994) and Waldman (1993) analyze a monopolist's compatibility decision between old and new products when the monopolist can make old units obsolete by introducing incompatible, new products. However, they assume that the monopolist has only two choices, non-compatibility and two-way compatibility. In their models, backward compatibility is always a better choice than non-compatibility. Therefore, they do not explain why the monopolist chooses non-compatibility rather than backward compatibility.

6 Conclusion

We have built a formal vertically-differentiated products model that allows the interaction of consumers' preferences and network effects. Within this framework, we studied a hardware firm's decision to make its products forward and backward compatible. The one-way compatibility decision dramatically changes the relationship among consumers-software-hardware and has critical effects on market adoption of new technology and the profits of a monopoly supplier of hardware.

There are several potential areas for extended research. We have analyzed these issues in the model with only three types of consumers. If we extend

this paper into a continuous-types model, we could get a more general market demand curve, which would be more useful for empirical work.

We have assumed that the hardware monopolist adopts the open licensing policy in the software market when she has property rights on the software market. However, there are several other strategies the monopolist can employ. It would be interesting to determine which strategies become optimal for the monopolist in which situations.

In this paper, no consumers initially had the old-generation system. Another case worth studying would be one in which a consumer's own past consumption pattern reveals her type. In other words, a consumer's type is related to whether or not she has the old system, as well as the number of software programs she has purchased. In this case, backward compatibility affects consumers differently along consumer types through their past consumption patterns.

Appendix (A)

Lemma 1: Suppose that b middle-type consumers and d high-type consumers purchase hardware N. At the free-entry equilibrium, every consumer owning hardware O buys v_o old generation software programs, each at price $\bar{p}_x$, where $\bar{p}_x = U_x(v_o, 0)$. Every consumer owning hardware N buys $y^m(\frac{F_N}{b+d})$ new-generation software programs, each at price $\frac{F_N}{b+d}$. When $y^h(\frac{F_N}{d})$ is higher than $y^m(\frac{F_N}{b+d})$, every high-type consumer purchases additional $(y^h(\frac{F_N}{d}) - y^m(\frac{F_N}{b+d}))$ programs, each at price $\frac{F_N}{d}$.

Proof. Let $[v_x,\ p_{x1},\ p_{x2},\ ...\ p_{xv_x}]$ denote the number and prices of old-generation software programs. Let $[v_y,\ p_{y1},\ p_{y2},\ ...\ p_{yv_y}]$ denote the number and prices of new-generation software programs in the free-entry equilibrium. We will find the number and prices of software programs satisfying the free-entry equilibrium condition. We sort software programs according to their prices, starting from the lowest to the highest: $p_{x1} \leq p_{x2} \leq .. \leq p_{xv_x}$ and $p_{y1} \leq p_{y2} \leq .. \leq p_{yv_y}$.

Claim One: When there are α old-generation software program, all of these α programs sell uniformly at price $U_x(\alpha, 0)$.

Step One. The lowest price, p_{x1}, must be equal or higher than $U_x(\alpha, 0)$.

Suppose not. Since the marginal utility $(U_x(\alpha, 0))$ is higher than the price (p_{x1}), the company could sell its program to all consumers owning hardware O even with a slightly higher price, which implies the pricing is not optimal.

Step Two. The highest price, p_k, must be $U_x(\alpha, 0)$.

If p_k is higher than $U_x(\alpha, 0)$, the program does not sell, since $p_k > U_x(\alpha, 0)$.

Step One and Step Two imply that all programs must be uniformly priced at $U_x(\alpha, 0)$.

By Assumption 2, software companies cannot recover its development cost, and there is no additional entry on the old-generation software side. Therefore, on the old-generation software market, every consumer owning hardware O buys v_x old generation software programs, each at price $\bar{p}_x$.

Claim Two: On the new-generation software market, the unique combination satisfying the free-entry equilibrium condition is $[v_y = y^m(\frac{F_N}{b+d}),\ p_{ni} = \frac{F_N}{b+d}$ for all $i = 1, 2,..y^m(\frac{F_N}{b+d})]$ when $y^h(\frac{F_N}{d}) < y^m(\frac{F_N}{b+d})$.

Step One. Suppose that some p_{ni} are lower than $\frac{F_N}{b+d}$. Then the programs whose prices are lower than $\frac{F_N}{b+d}$ cannot recover the development cost, F_n, even by selling their products to all potential consumers. So all prices must be higher than or equal to $\frac{F_N}{b+d}$.

Step Two. v_y must be $y^m(\frac{F_N}{b+d})$.

Suppose v_y is larger than $y^m(\frac{F_N}{b+d})$. The v_y-th variety cannot recover F_N, since $U_y(0, v_y; m) < \frac{F_N}{b+d}$ and $U_y(0, v_y; h) < \frac{F_N}{d}$.

Suppose v_y is less than $y^m(\frac{F_N}{b+d})$. Then the v_y-th variety can sell $b+d$ units with a higher price than $\frac{F_N}{b+d}$, since $U_y(0, v_y; m) > \frac{F_N}{b+d}$. The v_y-th variety makes a positive profit, which violates the free-entry condition.

Step Three. The $y^m(\frac{F_N}{b+d})$th variety cannot recover the fixed cost with a higher price than $\frac{F_n}{b+d}$ since $U_y(0, y^m(\frac{F_N}{b+d}); h) < \frac{F_N}{d}$ and $U_y(0, y^m(\frac{F_N}{b+d}); m) = \frac{F_N}{b+d}$. Therefore, the $y^m(\frac{F_N}{b+d})$th variety must be priced at $\frac{F_n}{b+d}$.

Step One and Step Three imply that all new-generation software programs are priced at $\frac{F_N}{b+d}$. Therefore, the combination $[v_y = y^m(\frac{F_N}{b+d}), p_{ni} = \frac{F_N}{b+d}\ for\ all\ i = 1, 2, ..y^m(\frac{F_N}{b+d})]$ is the unique one satisfying the free-entry equilibrium condition.

Claim Three: When $y^h(\frac{F_N}{d}) > y^m(\frac{F_N}{b+d})$, the unique combination satisfying the free-entry equilibrium condition is $[v_y = y^h(\frac{F_N}{d}),\ p_{ni} = \frac{F_N}{b+d}$ for $i = 1, 2,..y^m(\frac{F_N}{b+d})$ and $\mathrm{p}_{yi} = \frac{F_N}{d}$, for $i = y^m(\frac{F_N}{b+d}) + 1, .., y^h(\frac{F_N}{d})]$.

Step One. Suppose that some p_{ni} are lower than $\frac{F_N}{b+d}$. Then the programs whose prices are lower than $\frac{F_N}{b+d}$cannot recover the development cost, F_n, even by selling their products to all $b + d$ potential consumers. So all price must be higher than or equal to $\frac{F_N}{b+d}$.

Step Two. v_y must be $y^h(\frac{F_N}{d})$.

Suppose v_y is larger than $y^h(\frac{F_N}{d})$. The v_y-th variety cannot recover F_N, since $U_y(0, v_y; m) < \frac{F_N}{b+d}$ and $U_y(0, v_y; h) < \frac{F_N}{d}$.

Suppose v_y is less than $y^h(\frac{F_N}{d})$. Then the v_y-th variety can sell d units with a higher price than $\frac{F_N}{d}$, since $U_y(0, v_y; h) > \frac{F_N}{d}$. The v_y-th variety makes a positive profit, which violates the free-entry condition.

Step Three. When v_y is $y^h(\frac{F_N}{d})$, all programs that are ranked (based on price) between $y^m(\frac{F_N}{b+d}) + 1$, and v_y must be priced at $\frac{F_N}{d}$.

Middle-type consumers purchase at most $y^m(\frac{F_N}{b+d})$ programs since the prices are higher than or equal to $\frac{F_N}{b+d}$. Therefore, the programs that are ranked (based on price) between $y^m(\frac{F_N}{b+d}) + 1$ and v_y can sell at most d units. Therefore, the prices of these programs must be uniformly $\frac{F_N}{d}$.

Step Four. All programs that are ranked (based on price) between 1 and $y^m(\frac{F_N}{b+d})$ must be priced at $\frac{F_N}{b+d}$.

The $y^m(\frac{F_N}{b+d})$th variety can recover F_N by selling $b + d$ units at price $\frac{F_N}{b+d}$ or selling d units at price $\frac{F_N}{d}$. Suppose $y^m(\frac{F_N}{b+d})$th variety sells d units at price $\frac{F_N}{d}$. In this case,($y^m(\frac{F_N}{b+d}) - 1$)th variety can make profits more than F_N with selling $b + d$ units at price $U_y(0,\ y^m(\frac{F_N}{b+d}) - 1; m) > \frac{F_N}{b+d}$, which violates the free-entry equilibrium condition. Therefore, at the free-entry equilibrium, the $y^m(\frac{F_N}{b+d})$th variety must be priced at $\frac{F_N}{b+d}$. Since all prices are no less than $\frac{F_N}{b+d}$, all programs that are ranked between 1 and $y^m(\frac{F_N}{b+d})$ must be priced at $\frac{F_N}{b+d}$. Therefore, the combination $[v_y = y^h(\frac{F_N}{d}),\ p_{ni} = \frac{F_N}{b+d}$for $i = 1, 2,..y^m(\frac{F_N}{b+d})$ and $\mathrm{p}_{yi} = \frac{F_N}{d}$, for $i = y^m(\frac{F_N}{b+d}) + 1, .., y^h(\frac{F_N}{d})]$ is the unique free-entry equilibrium outcome. ■

References

A.Brandenburger and B.Nalebuff, Co-opetition. 1996

A.Brandenburger, "Power Play (A): Nintendo in 8-bit Video Games," 795-102, 1995, "Power Play (B): Sega in 16-bit Video Games," , 795-103, 1995, "Power Play (C): 3DO in 32-bit Video Games," Harvard Business School Publishing, 795-104, 1995.

Bensen, S. and Farrell, J. 1994, Choosing how to compete: Strategies and Tactics in Standardization, *Journal of Economic Perspective* Vol 8 Number 2 117-131

Blackstone, E. 1975, Restrictive Practices in the Marketing of Electron-Fax Copying Machines and Supplies: The SCM Corporation Case, *Journal of Industrial Economics* 23:189-202

Choi, J.P 1994, Network externality, compatibility choice and Planned obsolescence, *Journal of Industrial Economics* 42, 167-182

Choi, J.P 1996, Do converts facilitate the transition to a new incompatible technology? A dynamic analysis of converters, *International Journal of Industrial Organization*, October 1996, 825-835

Choi, J.P 1997, Herd Behavior, the Penguin Effect, and the Suppression of Informational Diffusion: Analysis of Informational Externalities and Payoff Interdependency, *Rand Journal of Economics*, Autumn 1997, 407-425

Chou, Chien-fu and Oz Shy, 1990 Network effects without network externalities, *International Journal of Industrial Organization* 8, 259-270.

Church, J., and N. Gandal, 1992 "Network effect, Software provision and Standardization," *Journal of Industrial Economics*, XL:85-104

Church, J., and N. Gandal, 1993 "Complementary Network Externalities and Technology Adoption", *International Journal of Industrial Economics*, 11; 239-260

Church, J., and N. Gandal, 1999 "Systems Competition, Vertical Merger and Foreclosure" *Journal of Economics & Management Strategy*, forthcoming.

David, P.A and Bunn, J. A., 1988 "The Economics of Gateway Technologies and Network Evolution: Lessons from Electricity Supply History', Information Economics and Policy, Vol. 3, pp165-202

Dranove, D., and N. Gandal, 1999 "From DVD vs. DIVX Standard War" Working Paper

Gandal, N., M. Kende, and R. Rob, "The Dynamics of Technology Adoption in Hardware/Software Systems: The case of Compact Disk Players," Sackler Institute of Economics Studies Working Paper 21-97.

Economides, N. 1989, Desirability of Compatibility in the Absence of Network Externalities, *American Economic Review* Vol. 79,1165-1181

Economides, N and S. Salop, 1992, Competition and Integration among Complements, and Network Market Structure, The Journal of Industrial Economics Volume XL Number 1 March.

Ellison, G. and D. Fudenberg 1999, The Neo-Luddite's Lament: Excessive Upgrades of Computer Software, *Mimeo*

Farrell, J and Shapiro, C. 1992, Standard setting in high definition television, Brookings Papers: Microeconomics 1992.

Farrell, J. and Saloner, G. 1985, Standardization, Compatibility and Innovation, *Rand Journal of Economics* Vol. 16, 70-83

Farrell, J. and Saloner, G. 1986, Installed base and Compatibility, *American Economic Review* Vol. 76, 940-955

Fudenberg, D. and J. Tirole 1998, Customer Poaching and Brand Switching, *Mimeo.*

Fudenberg, D. and J. Tirole 1998, Pricing to Deter Entry by the Sole Supplier of a Network Good, *Mimeo.*

Katz, M. L. and Shapiro, C. 1985, Network externalities, competition and compatibility, *American Economic Review* Vol. 75. 424-440

Katz, M. L. and Shapiro, C. 1986, Technology adoption in the presence of network externalities, *Journal of Political Economics* Vol. 94. 822-841

Katz, M. L. and Shapiro, C. 1992, Product Introduction with Network Externalities, *Journal of Industrial Economics,* Vol XL Number 1 March, 55-83

Katz, M. L. and Shapiro, C. 1994, System Competition and Network Effects, *Journal of Economic Perspective* Vol 8 Number 2 Spring, 93-115

Matutues, C. and Regibeau, P. 1988, Compatibility and Bundling of Complementary Goods in a Duopoly, *The Journal of Industrial Economics* Volume XL March.

McGahan Anita, "Philips' Compact Disc Introduction (A)" Harvard Business School Publishing, 9-792-035, 1993, "Philips' Compact Disc Introduction (B)" Harvard Business School Publishing, 9-792-036, 1991, "Philips' Compact Disc Introduction (C)" Harvard Business School Publishing, 9-792-037, 1991.

McGahan, A. M. "The Incentive Not to Invest: Capacity Commitments in the Compact Disk Introduction." In Research on Technological Innovation Management and Policy, vol. 5, edited by R. A. Burgelman and R. S. Rosenbloom. Greenwich, Conn.: JAI Press, 1994.

Waldman, M. 1993. A new Perspective on Planned Obsolescence. Quarterly Journal of Economics 108, 273-283.

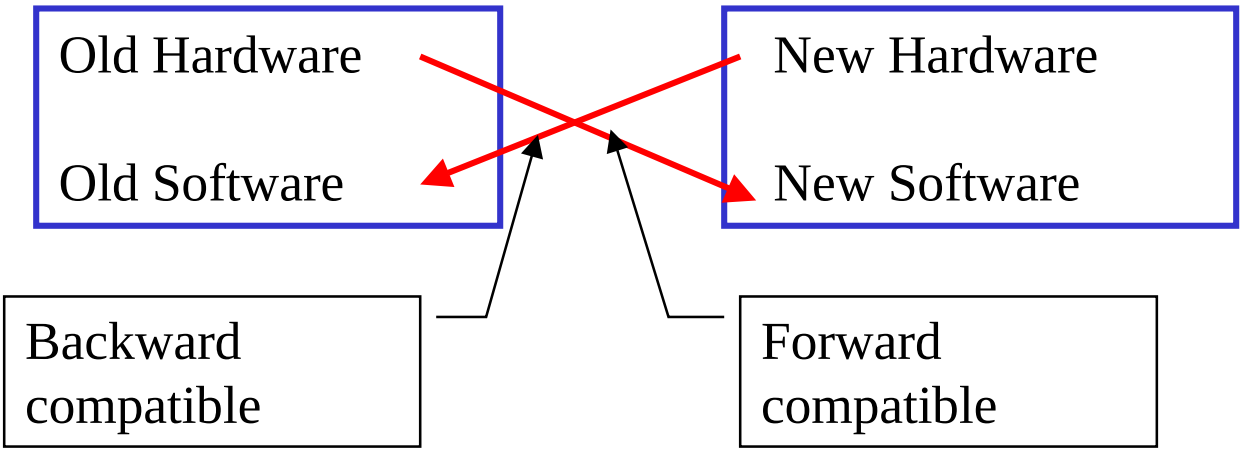

Figure 1.1 Compatibility between two generations of standards: Backward compatibility and forward compatibility.

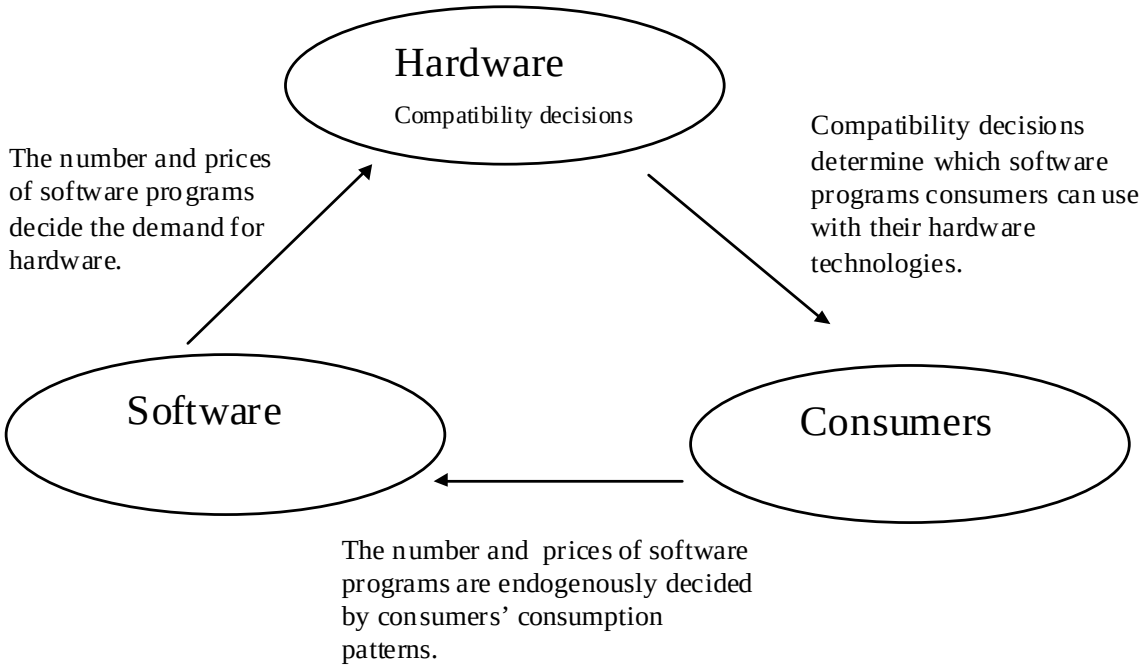

Figure 1.2. The relationships among hardware, software, and consumers.

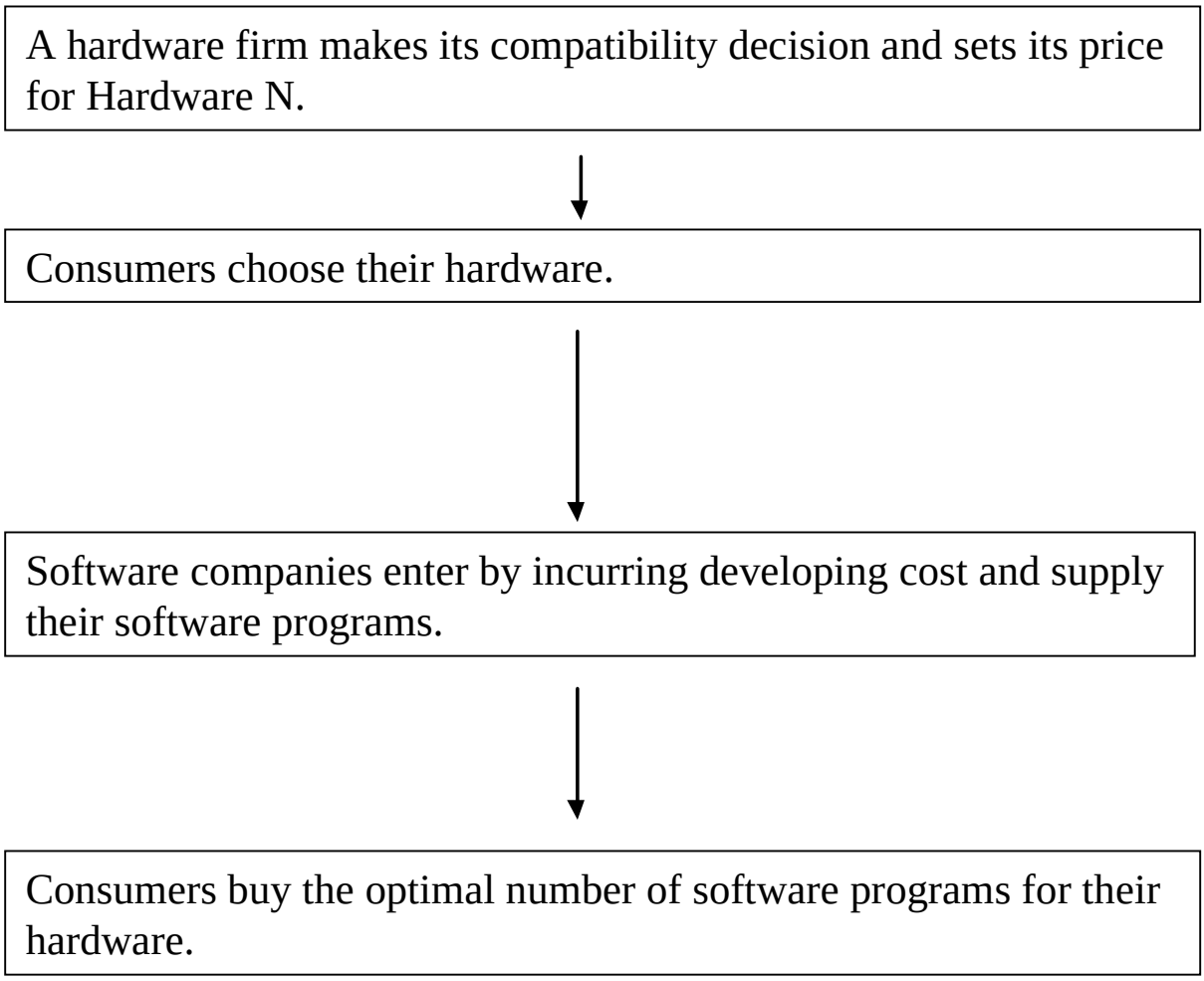

Figure 2.1. Timing of the game.

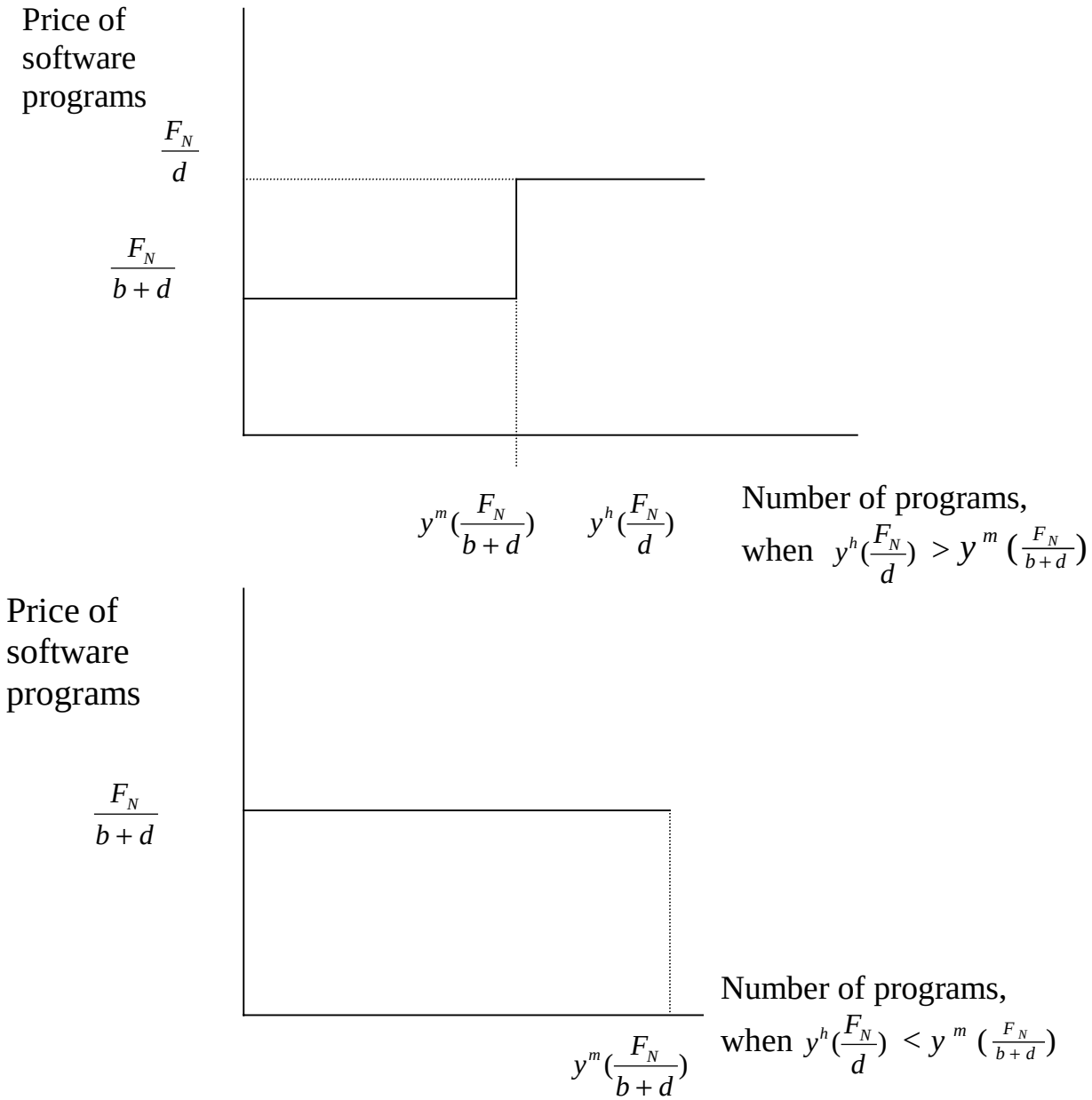

Figure 3.1. All consumers with hardware N buy $y^m(\frac{F_N}{b+d})$ programs each at $\frac{F_N}{b+d}$. When $y^h(\frac{F_N}{d})$ is larger than $y^m(\frac{F_N}{b+d})$, high type consumers buy additional $(y^h(\frac{F_N}{d}) - y^m(\frac{F_N}{b+d}))$ programs each at price $\frac{F_N}{b+d}$.

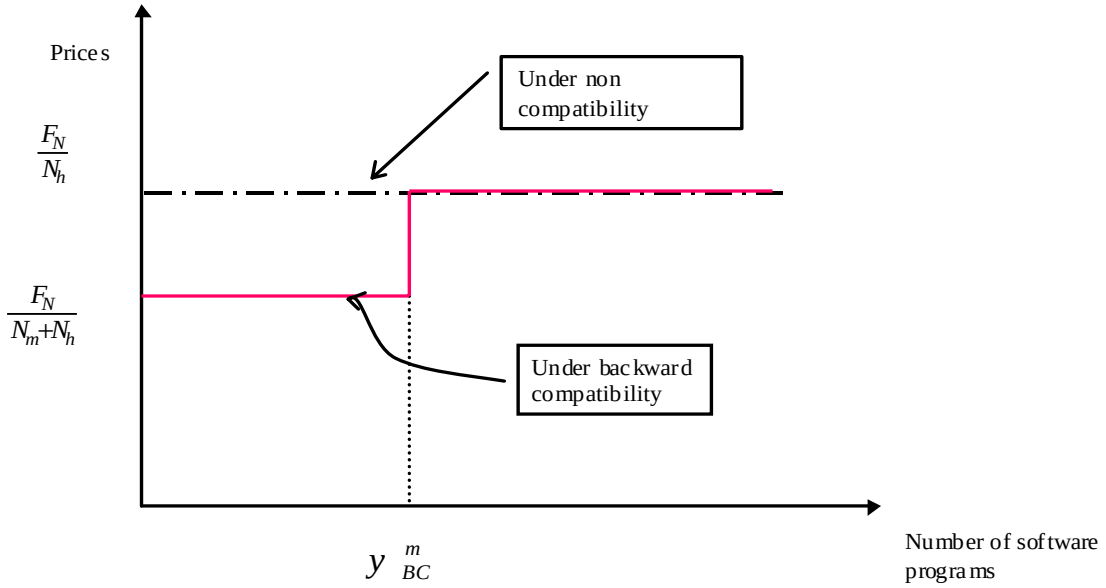

Figure 3.2A. Backward compatibility increases the consumer base and high-type consumers' surplus from hardware N.

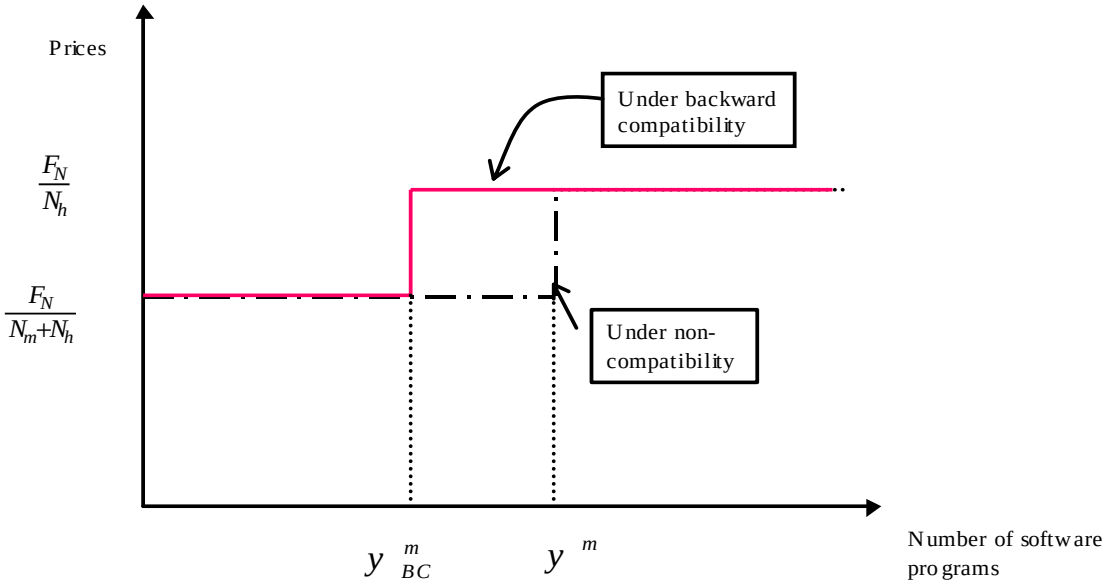

Figure 3.2B. Backward compatibility can decrease high-type consumers' surplus from hardware N.

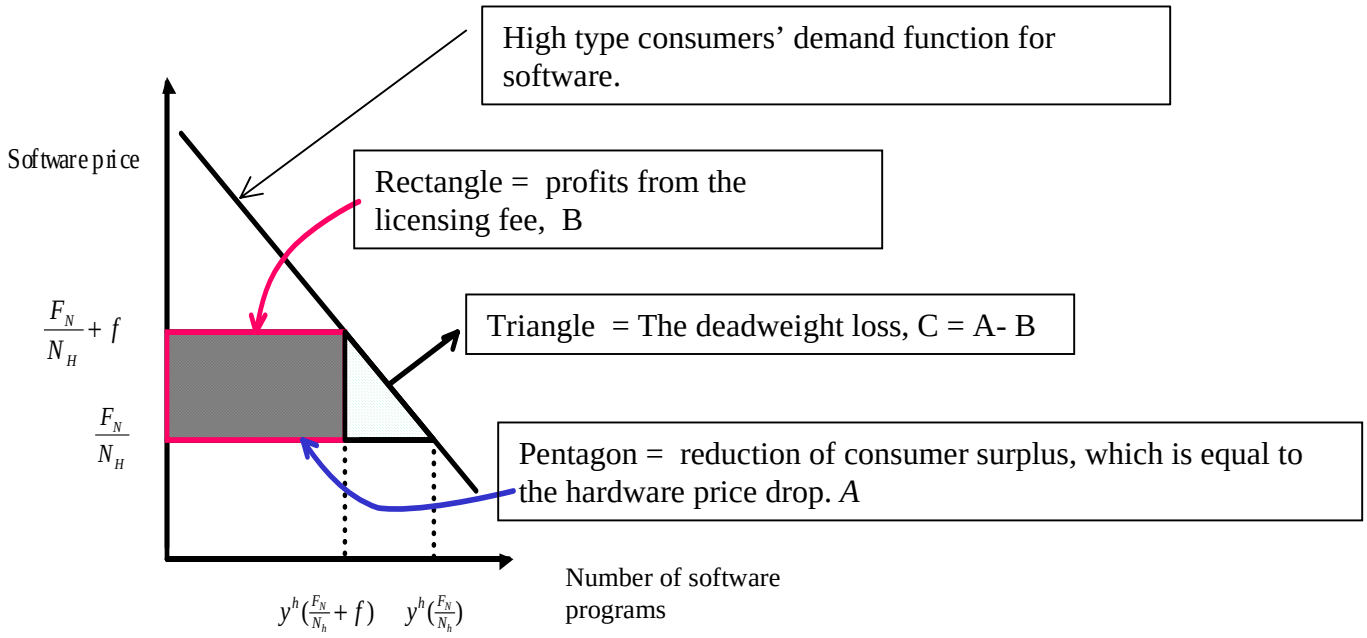

Figure 5.1. The gain of profits on the software market generated by a licensing fee is always less than the loss of profits on the hardware market when high-type consumers are the marginal type.

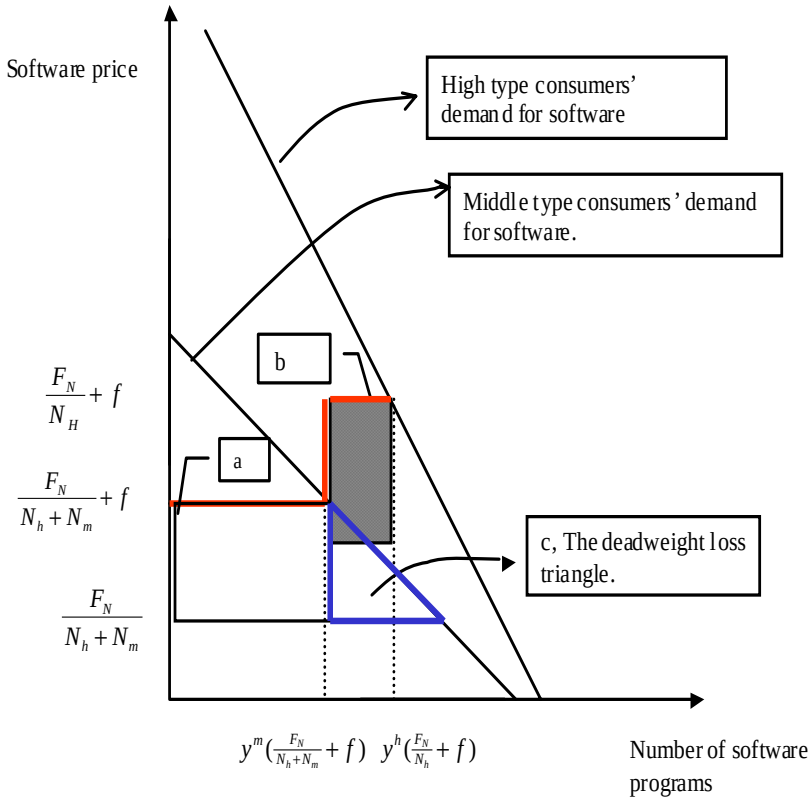

Figure 5.2. The gain of profits on the software market can be larger than the loss of profits on the hardware market. Therefore, the firm can increase its profitability with a positive licensing fee.