LICENSING, SOFTWARE SYNERGY, AND INDIRECT NETWORK EXTERNALITIES

by

M. KENDE*

95/46/EPS

* Assistant Professor of Economics, at INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

Licensing, Software Synergy, and Indirect Network Externalities

Michael Kende[1]

INSEAD

May 1995

Abstract

This paper examines the effectiveness of two common strategies used by a monopoly seller of the hardware component of a system. The first strategy consists of licensing the hardware technology to third-party firms, thus encouraging software firms to develop products for the licensed standard. The second strategy is for the hardware manufacturer to integrate into software production and to unilaterally commit to a high level of compatible software. Both strategies not only increase profits from hardware sales, but also increase consumer surplus by lowering hardware prices and increasing software availability.

Keywords: Network Externalities, Monopoly, Licensing.

[1] Correspondence: Michael Kende, INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France. Fax number 33 1 6072 4242, email address: Kende@Insead.Fr. I would like to thank Antonio Fatás, Neil Gandal, Rafael Rob, and Peter Zemsky for their input on this and previous drafts. All remaining errors are my own, of course.

1. Introduction

Consumer electronics products are often composed of both a hardware component and its complementary software. For consumers, this type of system implies indirect network externalities: the larger the installed base of hardware, the more software is available, increasing consumers' utility. For manufacturers with a monopoly over the hardware component, this implies adopting one of two strategies. The first consists of licensing the hardware technology to third-party firms, and the second of integrating into software production. This paper formalizes these strategies and shows that both not only increase profits by increasing software availability, but also increase consumer surplus.

The two strategies examined in this paper are ways for the monopolist to avoid the inertia which results from a type of chicken-and-egg dilemma facing consumers and software firms. The chicken-and-egg dilemma is that consumers would like to wait for firms to develop software before buying hardware, while software firms would like to wait for consumers to buy hardware before developing software. For both sides committing to the system requires investments which are at least partly system-specific, and therefore become sunk costs. Each side has an incentive to wait to avoid the risk of sinking money into an unpopular system. The two strategies explored in this paper solve the chicken-and-egg dilemma by increasing software availability before the system is introduced to consumers.

The first strategy examined in this paper is for the hardware producer to license the technology to at least one other firm. When a hardware monopolist licenses a proprietary hardware standard competition is introduced. It is hence not intuitively obvious that profits should increase, though many commentators take the consequence for granted. For instance, when Apple recently decided to begin licensing its proprietary operating system, many industry analysts felt that this move should have come earlier, before Microsoft's Windows established itself as the dominant alternative.[2] The question is how would earlier licensing have helped Apple? Licensing would have created competition between different hardware manufacturers, but the outcome of this competition could have been duplicated by Apple simply increasing production and lowering prices. The answer to the question therefore lies beyond merely creating competition. This paper's answer to the question is that licensing is a commitment to low future sales prices for hardware, and thus acts as a signal to software firms to support the standard, which can begin a virtuous circle in favor of the standard.

The importance of licensing lies in its ripple effects. Licensing contracts in this paper specify that the licensees pay a fixed royalty to the licensor for every unit shipped. The royalty payment is relatively low, and therefore, given the competition created by licensing, the final sales price of the hardware is expected to be low. These low expected prices signal to software firms a commitment to large future sales volumes. This in turn increases the likelihood of software development, which results in a greater variety of software for the given hardware. This increases demand for the hardware beyond the price effect of the licensing agreement. Therefore, the total increase in sales can offset the lower prices of hardware caused by licensing to increase profits for the licensor.

Economides (1995) and Kende (1995) have both studied issues related to the licensing strategy of this paper. In both of these papers a hardware standard is licensed in order to profit from network externalities. In Economides (1995), while there is a monopoly system producer as in this paper, the network externalities are direct. If the network effect is strong enough the monopolist can increase profits by inviting entry. In Kende (1995), while the network externalites are indirect, licensing is a strategic move designed to affect the competition between two differentiated standards. As in the present paper, licensing acts as a commitment to lower future hardware prices, inducing more software developers to support the licensed standard. The results of Kende (1995) show that licensing can increase the profits of the licensor at the expense of a non-licensing rival.

[2] "Anybody Wanna Clone a Mac?," *Business Week*, September 26, 1994.

This paper is also related to the second-sourcing literature (see Farrell and Gallini, 1988, and Shepard, 1987). In this literature consumers must incur a setup or adoption cost before price or quality is chosen by a monopolist. If the seller cannot contract on price or quality *ex ante* there is a dynamic consistency problem; *ex post* the firm would choose a relatively high price or low quality level. Second-sourcing may increase profits by ensuring *ex post* competition leading to lower prices or higher quality levels. As in the second-sourcing literature, in the present paper creating competition can increase profits. The difference with the second-sourcing literature is that here there are two players who must be induced to sink money: software vendors and consumers. Their actions are linked by the presence of indirect network externalities. In the present paper the hardware developer commits to low future prices via licensing in order to induce independent firms to incur the sunk cost of developing compatible software, thereby inducing consumers to purchase hardware.

Another strategy to solve the chicken-and-egg problem is for hardware manufacturers to integrate into software production. Integration provides the hardware monopolist with a direct means for increasing software availability before the system is introduced to consumers. This increased software availability increases the sales and profits of the integrated system manufacturer. An example of this strategy is Microsoft producing application software to complement its operating systems (which can be viewed as the "hardware" in the present analysis). Consumer electronics firms such as Sony are also integrating into software production by purchasing record companies and movie studios in an attempt to benefit from a synergy between hardware and software provision. When introducing a new audio or video hardware standard, these integrated firms can then insure software support from their own subsidiaries.

Papers related to the software provision strategy include Church and Gandal (1993a,b) and Kende (1994). In Church and Gandal (1993a) there are two differentiated hardware standards and a variety of software for each standard. They use the model to show the incentives of a hardware firm to vertically integrate into software production in order to favorably affect the outcome of competition between standards. In Church and Gandal (1993b) they examine the case where the vertically integrated system supplier no longer makes software compatible with the rival system. This can lead to monopolization of the hardware market. Kende (1994) uses a similar model to show the incentives of a hardware monopolist to unilaterally foreclose software competition by providing a closed system. In the present paper the monopolist provides software within an open system framework in order to increase sales of hardware.

The model is presented in Section 2. In this section the benchmark case is shown for when the hardware is not licensed and all software is produced by independent vendors. In Section 3 the equilibrium is derived for when the monopolist licenses the hardware standard to at least one other firm. Section 4 contains the results when the hardware manufacturer produces some software as well as the hardware. The welfare results are in Section 5. Finally, the conclusion is in Section 6.

2. Model

In this model consumers have unit demands for both hardware and a variety of software. The net utility function is as follows:

$$U = N^{\frac{1}{2}} + \upsilon - p - \sum_{i=1}^{N} \rho_i \qquad (1)$$

where N is the amount of software consumed, and v, the standalone benefit from consuming the system, is distributed uniformly between zero and one. Consumers benefit from an increased variety of software, but at a decreasing rate. The price of the hardware unit is p, and total expenditure on software equals $\sum_{i=1}^{N} \rho_i$. For simplicity I assume that software is symmetric, therefore expenditures on software equal $N\rho$. From the maximization of net utility one can show that the equilibrium software price is:

$$\rho = \frac{1}{2} N^{-\frac{1}{2}}. \tag{2}$$

No firms would deviate from this price. There is no incentive for software vendors to lower price because given the unit demand for software sales wouldn't increase. A higher price would reduce sales to zero because it would be greater than the marginal utility of the software to the consumer.

Substituting (2) into (1) gives the indirect utility function:

$$V = \frac{1}{2} N^{\frac{1}{2}} + \upsilon - p. \tag{3}$$

From this one can derive the demand curve facing the monopolist:

$$Q = 1 - p + \frac{1}{2} N^{\frac{1}{2}}. \tag{4}$$

The timing of the game is as follows. In the first stage the monopolist introduces the hardware component of a proprietary system standard. At this stage the monopolist could choose to implement either of the two strategies examined here: licensing the hardware or developing a selection of complementary software. This commitment is public; either all parameters of the licensing agreement or the variety of software are known to everybody. The effectiveness of these strategies is examined in Sections 3 and 4. In stage two software firms enter the market by developing a variety of software for the standard. Software development entails a fixed cost, *F*, and this investment, being hardware-specific, becomes a sunk cost. Competition between software firms is monopolistically competitive, with free entry until profits equal zero. Finally, in stage three all firms set prices and there are sales of hardware and software. The game is solved backwards to insure subgame perfection.

The hardware producer is assumed to have no costs. The profit function of the hardware producer is:

$$\pi = pQ. \tag{5}$$

Each software producer has the following profit function:

$$\pi_s = \rho\, Q - F \tag{6}$$

where F is the cost of developing the software. I assume throughout that $F > 1/4$ to guarantee that all equilibria exist.

2.1 Equilibrium

In this section the benchmark case is examined where the hardware monopolist makes no commitment to license the hardware or produce software. The timing of the benchmark game is the following. In the first stage the system is announced, in the second stage N software developers sink F into developing a unit of software, and in the third stage the price of hardware is set and consumers purchase hardware and software. This is solved backwards in order that the game is subgame perfect.

In the third stage the monopolist maximizes profits given the amount of software, N, which has been developed. The first order condition yields:

$$p = \frac{2 + N^{\frac{1}{2}}}{4} \tag{7}$$

and therefore

$$Q = \frac{2 + N^{\frac{1}{2}}}{4}. \tag{8}$$

In the second stage there is free entry until π_s equals zero. The equilibrium of this game is in the following proposition.

Proposition 1:

For $F > 1/4$, the amount of software developed equals:

$$N = \left(\frac{2}{8F-1}\right)^2.$$

The price of hardware equals:

$$p = \frac{4F}{8F-1},$$

and the total quantity of hardware sold is:

$$Q = \frac{4F}{8F-1}.$$

Profit for the hardware monopolist equals:

$$\pi = \left(\frac{4F}{8F-1}\right)^2.$$

Proof:

For $F > 1/4$, N is derived from the free-entry condition (6), given the price of software in (2). Substituting the result into (7) and (8) gives p and Q.

3. Licensing

The monopolist can choose to license the hardware standard to third-party firms in stage one as a means of increasing profits. It is shown below that the monopolist licenses the hardware standard at a low royalty rate as a commitment to lower hardware prices. By credibly committing to a lower price for hardware eventual sales are increased. Software developers are more likely to sink money into development, the more hardware is expected to be sold. Therefore, the price commitment made by licensing at a low royalty acts as a means of increasing software availability. This in turn increases hardware sales beyond the increase from the price effect alone, and the net result given below is that total profits rise.

The timing of the game is the following. In stage one the monopolist announces that there is at least one licensee and announces the licensing parameters. It is assumed that the licensor charges a per-unit royalty and that the licensor and licensee(s) produce undifferentiated goods. The monopolist effectively constrains her future actions by creating competition in this fashion.

Lemma 1:

By charging a royalty of *r* to the licensee(s), where *r* is relatively low, the monopolist is committing to a hardware price of $p=r$ for stage three.

Proof:

The monopolist and the licensee(s) sell undifferentiated goods. Neither has an incentive to raise the price above $p=r$ because they would lose all sales to the other. The licensee(s) won't lower the sales price below *r* because this would mean selling below cost. The monopolist won't set price under *r* either, because the *ex post* profit-maximizing price is shown to be greater than *r*. Given the amount of software that is available due to the licensing agreement, the monopolist would like to *raise* price in stage three, but is constrained by the competition resulting from licensing.

Given the results of lemma 1, in stage three the licensor and licensee(s) both set the price of hardware, p^L, equal to *r*. Therefore, the profits of the licensor are equal to $\pi^L = p^L Q(p^L)$, because the licensor earns revenues of p^L from every unit sold by either herself or the licensee(s).

In the second stage independent firms invest *F* in developing software. In light of lemma 1, software firms take the third-stage price of hardware as given at $p^L = r$. There is entry of software producers until the profits equal zero. Solving the zero-profit condition for the independent software vendors shows that the number of firms that enter equals:

$$N^L = \left(\frac{2(1-p^L)}{4F-1}\right)^2 . \tag{9}$$

In the first stage the monopolist maximizes total profits in the royalty rate, *r*. When determining the profit-maximizing royalty the monopolist takes into account the effect of the royalty on the number of software products developed in stage two, as shown in equation (9) above. In this stage the monopolist knows that, according to lemma 1, the eventual price of hardware equals the royalty rate, implying that total profits to be maximized are $\pi^L = p^L Q(p^L)$. The following proposition gives the equilibrium of this game.

Proposition 2:

With licensing, for $F > 1/2$ the number of software products developed is:

$$N^L = \left(\frac{1}{4F-1}\right)^2,$$

the price of hardware is:

$$p^L = \frac{1}{2},$$

the quantity of hardware sold is:

$$Q^L = \frac{2F}{4F-1},$$

and the profits of the monopolist are:

$$\pi^L = \frac{F}{4F-1}.$$

For $F \leq 1/2$ there is a corner solution. In this case $Q^L = 1$ and the price is:

$$p^L = \frac{1}{4F},$$

and profits equal:

$$\pi^L = \frac{1}{4F}.$$

Finally, the total variety of software when there is a corner solution is:

$$N^L = \frac{1}{4F^2}.$$

For all $F > 1/4$ profits are greater when the monopolist licenses the hardware standard.

Proof:

For $F > 1/2$ the solution is derived by simply maximizing in stage one the profits of the hardware producer to get the optimal royalty rate, and then substituting the result into (9) to get N^L. For $F \leq 1/2$ there is a corner solution. In this case the monopolist maximizes profits by committing via the royalty rate to a hardware price that leaves the marginal consumer (v=0) with zero utility, given the amount of software that is produced when $Q^L = 1$. It can be shown that, given N^L, in stage three the monopolist would like to charge a price higher than p^L, but is effectively constrained according to lemma 1 by the competition from the licensee(s). Finally, in either case, i.e. for all $F > 1/4$, profits are greater with licensing.

The monopolist, by licensing, commits to a low price for the hardware in order to induce more software manufacturers to enter the market before sales are made to consumers. The increased availability of software multiplies the effect of the lower hardware price on hardware sales, causing overall profits for the licensor to increase. For low enough software development costs, the increased sales of hardware results in a corner solution where the whole market is covered.

Licensing, as shown above, is an indirect means of increasing software availability before the introduction of the system to consumers. A more direct way is for the hardware monopolist to also produce software. This strategy is examined below.

4. Software Synergy

In this section the second strategy available to the monopolist, integrating into software production, is examined. By producing software the monopolist can unilaterally commit *ex ante* to a high level of software availability, thereby increasing eventual demand for hardware and ultimately profits. In this game again there are three stages. In the first stage the monopolist invests in producing a variety of software, N^m. The monopolist's cost of developing a type of software, *F*, is assumed to be identical to that for the independent software vendors. In the second stage there is entry of independent software producers. Independent software producers enter until their number, N^I, combined with the amount of software produced by the monopolist, yields zero profits. Finally, in stage three there are sales of both hardware and software.

The objective function of the monopolist producing both hardware and software is:

$$\pi^S = pQ(N^S) + N^M[\rho(N^S)Q(N^S) - F], \tag{10}$$

where $N^S = N^I + N^M$, and the S superscript refers to the synergy case.

The game is solved backwards. In stage three the monopolist maximizes profits in the price of hardware, taking as given the total amount of software available. This results in the following:

$$p^S = \frac{1 + \frac{1}{2}N^{S\frac{1}{2}} - N^M \rho}{2} \tag{11}$$

and

$$Q^S = \frac{1 + \frac{1}{2}N^{S\frac{1}{2}} + N^M \rho}{2}. \tag{12}$$

In stage two there is entry of N^I independent software producers, until, given N^M, profits equal zero. Solving the independent software producers' zero-profit condition for N^I yields:

$$N^I = \frac{2(1 - N^M + 12FN^M - 32F^2N^M + (1 - N^M + 8FN^M)^{\frac{1}{2}})}{(8F-1)^2} \tag{13}$$

and

$$N^S = \frac{2(1 - \frac{N^M}{2} + 4FN^M + (1 - N^M + 8FN^M)^{\frac{1}{2}})}{(8F-1)^2}. \tag{14}$$

Finally, in stage one the monopolist determines the amount of software to produce by maximizing the objective function (10), given (14). The solution to this is:

$$N^M = \frac{1}{(4F-1)^2}. \tag{15}$$

For $F \leq 1/2$ there is a corner solution because the relatively low software development costs leads to enough software created that the whole market is covered. In this case the monopolist in stage three sets the price for hardware to make the marginal consumer ($\upsilon = 0$) indifferent between buying or not buying, given the amount of software available. From (3) this yields:

$$p^S = \frac{1}{2} N^{S\frac{1}{2}}. \tag{16}$$

In stage two there is entry of the N^I independent software vendors. Finally, in stage one the monopolist determines the profit-maximizing level of software. Substituting (16) into the profit function (10) and maximizing yields that

$$N^M = \frac{1}{4F^2}.$$

Proposition 3:

When the monopolist commits to the above levels of software for any $F > 1/4$, then $N^I = 0$ and hardware sales, prices, and profits are exactly as in Proposition 2, the case with licensing. As in the case of the licensing strategy explored above, profits are greater than in the benchmark case.

Proof:

For $F > 1/2$, substituting N^M from (15) into (13) above yields the result that there is no independent software production, i.e. $N^I = 0$. Further substitution shows that p, Q and π are identical with the results of licensing. For $F \leq 1/2$, when the monopolist commits to N^M in stage one software profits equal zero. In stage two there are no entrants because they would depress prices and not be able to cover fixed costs. Substitution yields p, Q and π, which are again identical to the results of the licensing case.

Software production by the hardware-component monopolist is a strategy which increases profits. The monopolist increases software availability by producing a greater variety of software then would be developed otherwise. This acts to increase sales of hardware and also profits. The monopolist earns as much profit as in the licensing case by producing alone as much software as the independent software vendors would produce in the licensing case. By producing this optimal variety of software the monopolist forecloses the market to independent software competitors, making it a *de facto* closed system. However, the monopolist makes no net profits on software sales.[3] Software production is undertaken solely to increase hardware sales and profits.

5. Welfare Results

5.1 Consumer Surplus

Consumer surplus is:

$$\begin{aligned} CS &= \int_{\upsilon}^{1}[\frac{1}{2}N^{\frac{1}{2}} + n - p]dn \\ &= \frac{1}{2}N^{\frac{1}{2}}Q - pQ + \frac{1}{2} - \frac{\upsilon^2}{2} \end{aligned} \tag{17}$$

where Q=1-v.

[3] This result is consistent with the general results of charging a two-part tariff. Here the entry fee paid to the monopolist is the price of the hardware, and the software prices are similar to usage fees. Because consumers have homogeneous software demands, the optimal result consistent with two-part tariff theory is to charge a relatively high price approaching consumer surplus for the entry fee and a relatively low price approaching marginal cost for the usage fees.

5.2 Welfare Results

Welfare is assumed to be the addition of profits plus consumer surplus.

Consumer surplus in the benchmark case of Section 2 equals:

$$CS = \frac{8F^2}{(8F-1)^2},$$

and welfare equals:

$$W = \frac{24F^2}{(8F-1)^2}.$$

In the case of either type of monopoly commitment (superscripted with a *C*): licensing (Section 3) or software production (Section 4), for $F > 1/2$ consumer surplus equals:

$$CS^C = \frac{2F^2}{(4F-1)^2}$$

and welfare equals:

$$W^C = \frac{F(6F-1)}{(4F-1)^2}.$$

In the monopoly commitment case, for $F \leq 1/2$, when there is a corner solution, consumer surplus equals:

$$CS^C = \frac{1}{2},$$

and welfare equals:

$$W^C = \frac{2F+1}{4F}.$$

5.3 Welfare Comparisons

The following proposition gives the welfare comparisons:

Proposition 4:

For $F > 1/4$ both profits and consumer surplus are greater when the monopolist commits to licensing the hardware or producing a given amount of software. Therefore, these strategies both increase welfare.

Both strategies examined in this paper increase welfare because software availability rises while hardware prices fall and sales increase. In the case of the licensing strategy, the monopolist charges a relatively low royalty to the licensee(s). This acts as a credible commitment to a low sales price for hardware and higher sales volumes. Software manufacturers are therefore more willing to sink money into developing software, which in turn further increases the sales of hardware. The result is that the profits of the licensor rise. The monopolist can also directly commit to this software level by developing the software herself, with results identical to the licensing case. The increased level of software available for consumption using either strategy, along with the lower price for hardware, increases consumer surplus. Therefore either strategy increases welfare. This implies that the results of the benchmark case of using neither strategy imply suboptimal adoption of the standard.

6. Conclusion

When introducing new proprietary system standards, firms face a type of chicken-and-egg dilemma. Consumers may wait for complementary software to be developed before sinking money into the hardware. Software vendors, on the other hand, may wish to wait for consumers to begin adopting the hardware before sinking money into developing software for the new standard. The result of this may be suboptimal adoption of a new standard, from the point of view of all parties concerned: the hardware firm, software vendors, and consumers.

This paper shows the results of two common strategies adopted by hardware firms to solve the chicken-and-egg problem. The first strategy is to license the hardware technology to at least one other firm, creating competition and thereby committing to a low future sales price for the hardware. Software firms, confident that the lower prices will cause greater adoption of the hardware, are more willing to commit to this standard. With more software, even more hardware is adopted, and the result is that profits increase for the hardware producer. The second, more direct strategy used by hardware producers to increase software variety is to simply begin producing software. Although independent software vendors are effectively foreclosed from this market, the extra software production increases hardware sales and profits. In equilibrium the amount of software produced with both strategies is the same, and therefore the effects on the hardware producers profits are also the same. Finally, since the price of hardware falls and more software is consumed, both of these strategies increase consumer surplus.

These strategies are commonly seen in practice. One firm that utilizes both strategies is Microsoft. By making its operating systems (DOS/Windows) available for any IBM-compatible personal computer, Microsoft has effectively "licensed" its systems, and the resulting increased installed base has made firms more willing to develop software compatible with Microsoft's operating systems. The effectiveness of this strategy can best be seen by contrasting the results with those of Apple Computer, which until recently did not make its operating system available for use with compatible hardware. According to *Business Week* (September 26, 1994), Apple currently has only 10% of the PC market, and is licensing its operating system in order to increase the amount of compatible software available. Microsoft also gives an example of a firm integrating into software production by producing applications software to complement its operating systems. Anecdotal evidence, at least, suggests that this has also been an effective strategy.

There may be further explanations for using either of the strategies examined in this paper. Another explanation for licensing a hardware standard may be to increase hardware variety. In practice licensed products tend to be differentiated, either horizontally, by features, or vertically, by quality. Therefore, in addition to increasing software, licensing may be a way for the licensing firm to increase hardware variety by taking advantage of the skills and/or reputations of the licensees. By appealing to different categories of consumers, this of course would further increase the amount and variety of software available. Another explanation for integrating into software production may be simply that software sales are themselves profitable. In this paper the monopolist provides software in order to increase profits from hardware sales; net profits on software sales are zero. In some product categories there may be barriers to entry in software production, implying that software production by a hardware firm is in order .to increase profits from *software* sales. These points suggest a few of the possible avenues of future research.

Bibliography

Chou, C. and O. Shy, 1990, "Network Effects without Network Externalities," *International Journal of Industrial Organization*, 8, 259-270.

Church, J. and N. Gandal, 1993, "Integration, Complementary Products, and Variety," *Journal of Economics and Management Strategy*, 1, 651-675.

Church, J. and N. Gandal, 1993, "Equilibrium Foreclosure and Complementary Products," Sackler Institute of Economic Studies, Working Paper No. 3-93.

Economides, N., 1995, "Network Externalities, Complementarities, and Invitations to Enter," *The European Journal of Political Economy*, forthcoming.

Farrell, J. and N. Gallini, 1988, "Second-Sourcing as a Commitment: Monopoly Incentives to Attract Competition," *Quarterly Journal of Economics*, 103: 673-694.

Katz, M.L. and C. Shapiro, 1986, "Technology Adoption in the Presence of Network Externalities," *Journal of Political Economy*, 94: 822-841.

Kende, M., 1994, "Profitability under an Open versus a Closed System," INSEAD mimeo.

Kende, M., 1995, "Licensing and the Battle between Standards," INSEAD mimeo.

Shepard, A., 1987, "Licensing to Enhance Demand for New Technologies," *Rand Journal of Economics*, 18: 360-368.