Platform Neutrality and Content Quality: The Impact of App Stores’ Ranking Policies on App Quality

Jan Krämer* Oliver Zierke†

First version: August 15, 2017 This version: March 12, 2019

Abstract

We investigate how a mobile app store’s ranking policy affects app quality and welfare by means of a game-theoretic model with a monopoly app store and two competing app developers. The app store can either list apps according to their quality (quality-based ranking), or sell the first position in an auction (sponsored ranking). App developers differ in their efficiency to produce quality apps. We show that a quality-based ranking yields an unambiguously higher expected app quality and consumer surplus only if the app store can accurately assess the quality of apps, and if developers are relatively similar in their efficiency.

1 Introduction

Mobile app stores (in the following “app stores”), such as Google’s Play Store or Apple’s App Store have become an essential facility of the Internet ecosystem. For example, in 2016 US consumers spent about 2 hours and 51 minutes each day using their mobile device (smartphone or tablet), which is almost twice as much as in 2013; and about 60% of their digital media time using mobile apps, 16% more than in 2013 (comScore, 2017). 97% of the mobile devices run either the iOS (Apple) or Android (Google) operating system (comScore, 2017), which again use their respective app store to control which apps are available to users.

*Chair of Internet and Telecommunications Business, University of Passau, Dr.-Hans-Kapfinger-Str. 12, 94032 Passau, Germany. Email: jan.kraemer@uni-passau.de.
†Chair of Internet and Telecommunications Business, University of Passau, Dr.-Hans-Kapfinger-Str. 12, 94032 Passau, Germany. Email: oliver.zierke@uni-passau.de.
Arguably, if an app is not listed in an app store, it virtually does not exist for users.\footnote{Users could circumvent the app store by downloading apps directly from developers’ websites or by jailbreaking their phone. However, installing apps from outside the app store either requires a fair amount of technical knowledge and runs the risk of losing the warranty of the device (in case of jailbreaking), and/or it bears additional security risks. It is therefore only a feasible option for a small minority of the users.} In this sense, the operators of app stores are important digital gatekeepers that operate a two-sided market, which intermediates the exchange of apps between app developers, on the one side, and app users, on the other side of the market. As is typical for two-sided markets, app stores exhibit strong cross-side network effects. That is, more users attract more app developers, and vice versa, which reinforces a dominant position of an app store, constitutes effective entry barriers for competing app stores, and explains why currently only two app stores (each targeting slightly different user groups) have significant scale. In other words, large app stores enjoy significant market power. In a recent antitrust case against Google, this has also been confirmed by the European Commission from a legal point of view (European Commission, 2018). Apple’s App Store, for example, is estimated to have generated USD 11.5 billion in annual revenue for Apple in 2017 (Jones, 2018), and App Store purchases have increased by roughly 40% annually in the years from 2013 to 2016 (Ovide, 2017), currently amounting to USD 38.5 billion (Nelson, 2018). Both Apple and Google typically command a 30% revenue share from the developers in their respective app stores (Bergen, 2016), which is also an expression of the market power that they enjoy.

At the same time, in a given app store, the competition for users by competing app developers is very strong. As of March 2018, about 2 million and 3.6 million apps are available in Apple’s App Store and Google’s Play Store, respectively (Statista, 2018). Therefore, gaining prominence in the app store is of critical business importance for app developers.

In order to provide prominence to selected apps, both app stores traditionally use different rankings, which are typically based on app performance measures (e.g., the most downloaded apps, the most successful apps with respect to in-app-purchases) or based on editorial choice (e.g., the newest apps, the pick of the day). However, it is important to note that the editorial selection is made by the app store without financial compensation by the app developers,
and in an effort to promote apps that the app store deems to be of good quality and to be valuable to users. This is in the best interest of the app store, as it demands a revenue share from apps, and thus has an incentive to promote those apps that are likely to be used more often and thus generate higher revenues (Ahn, 2017).

In addition app stores allow users to search for apps directly. Before 2015, the search results would only display the so-called organic search results, i.e., apps were displayed and ranked based on objective characteristics, such as their quality. However, in 2015 Google (Siliski, 2015), and in 2016 Apple (Vincent, 2016) have begun to introduce sponsored search, allowing app developers to place bids in an auction for a more prominent placement on top of the organic search results (Villanueva, 2016).

Allowing apps to be ranked in the top position in return for financial compensation, but to a large degree irrespective of their quality, marks an important shift in the apps stores’ policy, because app developers can now simply buy prominence in the app store. Empirical research has found that rankings have a significant causal effect on consumer choice (see, e.g., Ursu, 2018). For example, in 2018, a slight change in the PlayStore’s ranking algorithm has led to a drop in downloads of some apps in the range of 70-90% (Lanier, 2018). Therefore, it is evident that apps that are ranked higher will receive more demand, everything else being equal.

Under a quality-based ranking, this provides app developers with strong incentives to invest in app quality, so that their apps appear high in the search results. On the one hand, this raises concerns that under a sponsored ranking app developers may rather invest in prominence than quality of their apps. On the other hand, a quality-based ranking, and the incentives that it provides to invest in quality, may be prone to the app store’s ability to assess app quality accurately, whereas the bids of the app providers under a sponsored ranking could be a less noisy signal for app quality and may therefore be preferred by quality app developers (D’Onfro, 2015; Balakrishnan, 2016).
In this paper we investigate exactly these trade-offs and highlight which impact the shift from a (pure) quality-based ranking to a (pure) sponsored ranking is likely to have on app quality and welfare. To this end, we study a stylized scenario with a monopolistic app store and two app developers that compete for users in a game-theoretic model. App developers differ in their efficiency to produce quality apps, and each developer has listed exactly one app in the app store. The app store demands a profit share from each developer. Users are more likely to download the top listed app. We find that the incentives of developers to invest in app quality are governed by three main effects. (1) Under a quality-based ranking, developers have a higher incentive to invest in app quality than under a sponsored ranking, everything else being equal, because they want to win the quality-based competition for the top position. This is denoted as the competition effect. It is strongest when app developers are of similar efficiency. (2) However, under a quality-based ranking, the app store has some uncertainty in determining app quality and may therefore erroneously list the lower quality app in the top position. This denies the higher quality app some economies of scale, which leads to a reduction in quality at the high end. In reverse, the lower quality app is granted additional economies of scale, which leads to an increase in quality at the low end, everything else being equal. This is what we denote as the scale and precision effect. It is also due to the scale effect that the expected spread in app quality between the low and high efficiency developer is lower under a quality-based ranking. (3) Due to its ability to elicit an additional payment from developers, the app store demands a lower revenue share under a sponsored ranking than under a quality-based ranking. A lower revenue share provides app developers with a higher incentive to invest in app quality. This gives rise to a revenue-share effect, which is stronger under a sponsored ranking.

Therefore, we find that the overall effect of a sponsored ranking on app quality is ambiguous. A quality-based ranking unambiguously leads to higher app qualities than a sponsored ranking only if developers are similar in efficiency and the app store’s ability to accurately
assess app quality is high. However, a sponsored ranking always leads to a higher spread in expected app qualities between the two developers.

We also investigate the case where the app store is biased and/or vertically integrated with one of the app developers, and always promotes this developer in the ranking. Whereas a biased ranking leads to unambiguously lower app qualities than a sponsored ranking, vertical integration yields a boost in the revenue-share effect, which can ultimately increase app quality. Even if the low efficiency app developer is vertically integrated and always promoted, the average expected quality may be higher than under a quality-based or sponsored ranking.

2 Related Literature

Our model relates to several streams of the literature. First and foremost, our model is related to models on biased intermediation, e.g., in the context of booking or search platforms. The paper that is closest to ours is by De Cornière and Taylor (2016), who study in a general setting how biased intermediation, predominantly in the context of a vertically integrated product or service, affects firms’ investment incentives. Our set up is related to their congruent payoffs model, in which they find that the vertically integrated firm, which is recommended by the platform, has a stronger incentive to invest (e.g., in quality) than the non-recommended firm. This is similar to our finding in the case of vertical integration and a biased ranking. However, unlike De Cornière and Taylor (2016), we assume that content providers (app developers) are heterogeneous with respect to their costs to invest in quality, and we focus on the case where none of the app developers is vertically integrated with the platform. Moreover, in contrast to De Cornière and Taylor (2016) we consider the platform as a strategic player, who can influence the developers’ investment incentives also through its strategic choice of the revenue share.

Second, our model is loosely related to models of consumer search (e.g., Varian, 1980), and more specifically to the burgeoning literature on directed consumer search in the context of
search engines (e.g., Armstrong et al., 2009; Athey and Ellison, 2011; White, 2013; Hagiu and Jullien, 2014). Although, in contrast to this literature, we are not interested in the specific mechanisms of consumer search per se, we borrow from it the notion that consumers can be divided into two subgroups: Whereas one group of users tends to follow a store’s and search engine’s recommendation (e.g., a ‘sales’ promotion, or a sponsored search link), the other group does not. A key difference between the literature on consumer search and our model is that the former assumes that the distribution of ‘quality’ among content providers (to whom search is directed) is exogenously given and not influenced by the search mechanism. Instead, we focus on the very fact how a sponsored recommendation affects content providers’ incentives to invest in the quality, and consequently the distribution of content quality.

Third, our investigation is related to the debate on ‘platform neutrality’, which is currently very topical among EU policy makers (see, e.g., CNNum, 2014; European Commission, 2017; Krämer and Schnurr, 2018; Dillet, 2018; ARCEP, 2018). In this context, ‘neutrality’ means any unreasonable discrimination among content providers, that is, discrimination that is not based on objective (quality) standards, but purely on commercial considerations. From a scientific view, Easley et al. (2018) note that the same issues that have been debated in the context on the net neutrality debate at the infrastructure level, can also be debated at the software level, i.e., in the context of dominant software platforms. They highlight app stores as an example for such dominant platforms and argue that the pay-for-prominence scheme that is implemented by a sponsored ranking is akin to a pay-for-priority scheme that has been condemned under net neutrality regulation. Evidently, selling the top ranked position to the app developer that bids the most to be in this position, or simply granting the top position to one’s own app, is a form of non-quality-based discrimination. However, there are also remarkable differences between “net neutrality” and “app store neutrality”. At the infrastructure level, different apps have different technical requirements with respect to the Quality of Service (QoS). Therefore, a QoS-based discrimination of apps, which could be implemented through a pay-for-priority pricing scheme, can be welfare enhancing, as it alle-
viates congestion for the most congestion-sensitive apps (see, e.g., Krämer and Wiewiorra, 2012). In the context of software platforms this reasoning does not apply, because there is no technical reason why one app would require more prominence than another app. An app does not technically perform better, just because it is listed higher. Thus, the only objective standard for the ranking of similar apps can be the quality of an app. Based on this view, there may even be better reasons to impose neutrality regulation on software platforms than on infrastructure platforms, as is done with net neutrality. However, our results show that this conclusion may be premature, because in many cases a ban of a sponsored ranking may not be welfare enhancing.

3 Baseline Scenario: Quality-Based Ranking

3.1 The Model

Consider a market with two app developers, each of which develops exactly one app. The developers differ in their efficiency to produce a quality app. In particular, we assume that the cost function of the less efficient developer, \( L \), to produce an app of quality \( q_L \), is \( C_L = q_L^2 \); and that the cost function of the more efficient developer, \( H \), is \( C_H = k q_H^2 \) with \( k \in (0, 1] \). Thus, we obtain that the \( L \) developer incurs (weakly) higher costs for producing a certain quality level than the \( H \) developer. Note that \( \frac{\partial C}{\partial q_i} > 0 \) and \( \frac{\partial^2 C}{\partial q_i^2} > 0 \), such that producing quality improvements becomes more difficult for higher levels of \( q_i, i \in \{ L, H \} \).

Both developers offer their app through a monopolistic app store (i.e., an app store with significant market power) that ranks the apps according to some ranking algorithm. We assume that there exists a unit mass of users and that all users have to use the app store in order to retrieve apps. Every user downloads exactly one of the two available apps. In

\[ \text{By quality, we summarize all relevant vertical characteristics of an app, i.e., the characteristics on which users agree that they make an app 'better'. At the same time, we abstract from differences in horizontal characteristics, which are subject to users' tastes. Although rankings and search results usually have to consider both dimensions, within a given user's 'taste profile' they should rank those apps higher that have a higher quality.} \]
line with the findings of the literature on directed consumer search and with the empirical literature on rankings, we assume that the app that is listed in the top position is downloaded more often than the app in the second highest position. Specifically, we assume that the app in the top position receives \( n_1 \) downloads, and the app in the second highest position \( 1 - n_1 \) downloads, with \( n_1 \in (1/2, 1) \).

We further assume that the app store may not be able to observe quality perfectly, and may therefore make a mistake when applying a quality-based ranking algorithm, such that the app of lower quality is erroneously ranked at the top. We denote the app store’s precision in determining quality by \( \gamma \in [0.5, 1] \). Altogether, under a quality-based ranking a developer \( i \) attains a number of

\[
N_1 = \gamma n_1 + (1 - \gamma) (1 - n_1)
\]

downloads if it provides a higher quality higher than its competitor, and

\[
N_2 = (1 - \gamma) n_1 + \gamma (1 - n_1)
\]

downloads if it provides a lower quality lower its competitor. In order to break ties, we assume that the app store promotes the \( H \) developer with probability \( \gamma \) when both qualities are equal.

In line with empirical evidence (see, e.g., Ahn, 2017), we make the reasonable assumption that apps of higher quality are used more often. For simplicity, we assume that app usage increases linearly with the quality of the app. Thus, total usage of the developer \( i \)’s app amounts to

\[
D_i(q_i, q_{-i}) = \begin{cases} 
N_1 q_i & \text{for } q_{-i} < q_i \\
N_2 q_i & \text{for } q_{-i} > q_i, 
\end{cases}
\]  

where \( q_{-i} \) is the quality of the competitor of developer \( i \). While apps are free to download for users, app developers receive a gross revenue per usage of their app, e.g., through advertisements or in-app purchases. Therefore, gross revenues of the developers can be written
as
\[ R_i = q_i \left[ F_{-i}(q_i) N_1 + (1 - F_{-i}(q_i)) N_2 \right], \quad (2) \]
where \( F_{-i}(q_i) = \text{prob}(q_{-i} < q_i) \), i.e., \( F_{-i} \) is the cumulative distribution function (CDF) of the competitor’s quality choice. Furthermore, the app store charges a share \( s \) of each app developer’s gross revenues. Thus, we can write an app developer’s profit function as
\[ \Pi_i = (1 - s) R_i - C_i, \quad (3) \]
and the app store’s profit function as
\[ \Gamma = s \sum_i R_i. \quad (4) \]

The timing of the game is then as follows:

1. **Stage** The app store announces its revenue share \( s \).

2. **Stage** The app developers \( H \) and \( L \) set their qualities \( q_L \) and \( q_H \), and the app store ranks the developer with the highest quality first with probability \( \gamma \).

3. **Stage** Users choose which app to download, observe the downloaded app’s quality and use it accordingly.

### 3.2 Equilibrium Derivation

We solve the game through backwards induction, beginning in Stage 3.

**Stage 3:** Given the developers’ quality choice in Stage 2 and the revenue share in Stage 1, app usage and the developer’s profits are readily given by equations (1) and (3), respectively.

**Stage 2:** The analysis of Stage 2 entails two cases. First, if app developers efficiency is relatively similar, both compete in qualities for the top position. As we will show below, in
this case there does not exist a pure strategy equilibrium in qualities, but a unique mixed strategy equilibrium. Second, if developer’s efficiency is very different, then developers do not effectively compete for the first position anymore and there exists a unique pure strategy equilibrium in qualities.

**Lemma 1** (Equilibrium Quality Choice under Quality-Based Ranking). *Under a quality-based ranking, there exists a unique pure strategy equilibrium when \( k \leq \tilde{k} \), and a unique mixed strategy equilibrium when \( k > \tilde{k} \), with \( \tilde{k} = \frac{N_1}{N_1 + \sqrt{N_1^2 - N_2^2}} \). In the pure strategy equilibrium, app developers \( L \) and \( H \) choose quality levels of

\[
q_L = \frac{(1 - s) N_2}{2}, \\
q_H = \frac{(1 - s) N_1}{2 k}.
\]

In the mixed strategy equilibrium, app developers \( L \) and \( H \) choose expected quality levels of

\[
E[q_L] = F_L(q_H) q_L + \int_{q_H}^{\bar{q}_L} f_L(q_L) q_L d q_L, \\
E[q_H] = F_H(q_H) q_H + \int_{q_H}^{q_L} f_H(q_H) q_H d q_H,
\]

whereby developer \( L \) (\( H \)) chooses its minimum quality \( q_L \) (\( q_H \)) with probability \( F_L(q_H) \) (\( F_H(q_H) \)), and otherwise chooses a quality level between \( q_H \) and \( \bar{q}_L \) according to the cumulative distribution function \( F_L \) (\( F_H \)) with corresponding density function \( f_L \) (\( f_H \)). The cumulative distribution functions as well as minimum and maximum quality levels are given
by:

\[
F_L = \frac{(1-s)q_L N_1 - k\bar{q}_L^2 - ((1-s)q_L N_2 - k\bar{q}_L^2)}{(1-s)(N_1 - N_2)q_L}
\]

\[
F_H = \frac{(q_H - \bar{q}_L)^2}{(1-s)(N_1 - N_2)q_H}
\]

\[
\bar{q}_L = \frac{1}{2}
\]

\[
\bar{q}_H = \sqrt{\frac{(1-s)\bar{q}_L N_1 - k\bar{q}_L^2}{k}}
\]

\[
\bar{q}_L = \frac{(1-s)(N_1 + \sqrt{N_1^2 - N_2^2})}{2}
\]

While the complete proof of this lemma is relegated to the Appendix, in the following we provide some more intuition and details. At first, it is important to see that the \(L\) developer’s rational quality choice is bounded from below and above. It is bounded from below, because \(L\) can always choose not to compete in the quality-based ranking and to settle with the second position. We denote this as \(L\)’s outside option, and in this case it would receive a profit of \(\bar{\Pi}_L = (1-s)q_L N_2 - C_L\). However, \(L\) will still choose a positive quality level in this case, because otherwise app usage, and hence its revenues, would be zero. Specifically, maximization with respect to \(q_L\) yields a lower bound of \(q_L = \frac{(1-s)N_2}{2}\), which corresponds to an outside option profit of \(\bar{\Pi}_L = \bar{q}_L^2\). In reverse, this means that \(L\) will only choose quality levels that provide an expected profit that is at least as high as in the outside option. In other words, if developer \(L\) enters the quality competition, its expected profit in the top position, i.e., \(\bar{\Pi}_L = (1-s)\bar{q}_L N_1 - C_L(\bar{q}_L)\), must be at least as high as in the outside option, i.e., \(\bar{\Pi}_L \geq \bar{\Pi}_L\). This allows to determine the upper bound on \(L\)’s quality choice \(\bar{q}_L = \frac{(1-s)(N_1 + \sqrt{N_1^2 - N_2^2})}{2}\).

Based on these insights one can now derive the rational quality choices for developer \(H\). Assume for a moment that \(H\) expects to always achieve the first position. Then, \(H\) would choose a unique profit maximizing quality level of \(q^m_H = \frac{(1-s)N_1}{2k}\). Evidently, if \(q^m_H \geq \bar{q}_L\) then \(H\)’s profit maximizing quality choice will indeed always grant it the top position in the quality-based ranking and \(H\)’s expectations are fulfilled. This gives rise to the pure strategy equilibrium in which \(H\) chooses a quality level of \(q^m_H\) and \(L\) a quality level of \(q_L\), i.e., where
both developers do not effectively compete for the top position. This case occurs if $H$’s efficiency is much larger than that of $L$, i.e., for values of $k \leq \tilde{k}$. However, if $q^m_H < \bar{q}_L$, then developers do compete for the top position and a pure strategy equilibrium does not exist.\(^3\) Consequently, both $H$ and $L$ would choose from a range of possible quality levels according to some probability distribution function $F_i$ in a mixed strategy equilibrium. We have already determined the bounds of the feasible quality levels for developer $L$ in such an equilibrium. The corresponding bounds for $H$ are now derived in a similar fashion. First, note that $H$ can always (just) outbid $L$ by choosing $q_H = \bar{q}_L$. We denote this as the outside option of $H$ in which it makes a profit of $\tilde{\Pi}_H = (1 - s) \bar{q}_L N_1 - C_H(\bar{q}_L)$. Consequently, $H$ would never choose a quality level that entails a lower profit than $\tilde{\Pi}_H$. This means that in equilibrium $\Pi_H \overset{!}{=} \tilde{\Pi}_H$, where $\Pi_H$ is given by (3). Solving this equality yields $F_L$. Likewise, we can determine $F_H$ by solving the equation $\Pi_L \overset{!}{=} \tilde{\Pi}_L$. Finally, we can now determine the lower bound of $H$’s quality choices as the lowest feasible quality level prescribed by the quality distribution functions $F_H(q)$ and $F_L(q)$. More precisely, in the Appendix we show that $q_H = \sqrt{\Pi_H/k}$. Therefore, $q_H$ is chosen by developer $H$ with a probability of $F_H(q_H)$ and $q_L$ by developer $L$ with probability $F_L(q_H)$.

\[\text{Figure 1: Distribution of } q_L \text{ and } q_H \text{ in the mixed strategy equilibrium under a quality-based ranking. Numerical example derived for } s = 0.5, n_1 = 0.75, \gamma = 1, k = 0.75.\]

\(^3\)This case corresponds the analysis of all pay auction under full information (see Riley, 1988). See the Appendix for details on the uniqueness of the mixed strategy equilibrium.
The resulting distribution of equilibrium qualities in the mixed strategy equilibrium is exemplified in Figure 1. It can be seen that the quality choices of the $H$ developer first-order stochastically dominate those of the $L$ developer.

**Stage 1:** Finally, in the first stage, the app store selects the revenue share, $s$, which it commands from the app developers. Thereby, it anticipates how its choice of $s$ will affect the developers’ subsequent quality choice. Since the app store is a monopolist, $s$ is determined by maximization of the app store’s profit as follows:

$$s = \arg \max_s \Gamma = \frac{1}{2}. \quad (5)$$

## 4 Sponsored Ranking

### 4.1 The Model

We now assume that developers compete for the first position in the ranking by submitting payments, $r_i$. The first position will be awarded to the developer that has submitted the highest payment. In other words, the first position is sold by means of a first-price all-pay auction.\footnote{The assumption of an all-pay auction mimics very well a market in which a potentially very large group of developers compete for the first position. In practice, often position auctions (Varian, 2007), also known as generalized second-price auction (Edelman et al., 2007) are employed, where the highest bidder attains the highest ranking position and has to pay the amount that the second highest bidder has bid. The second highest bidder attains the second position and pays the amount that the third highest bidder has bid, and so on. When there is a large group of bidders, each bidder will approximately have to pay its bid - like in a first-price all pay auction. Thus, our assumption allows us to mimic market outcomes in a large developer market by only considering the interaction of two developers. Moreover, this set-up is the most comparable to the quality-based ranking where developers effectively also play an all-pay auction (in qualities) in order to attain the first ranking position.} Moreover, in contrast to the quality-based ranking in Section 3, there exists no uncertainty in the determination of the first position, because payments are perfectly...
observable by the platform, i.e., $\gamma = 1$. This means that

\[
N_1 = n_1 \\
N_2 = (1 - n_1).
\]

Therefore, under a sponsored ranking regime an app developer’s profit is

\[
\Pi_i = (1 - s) R_i - C_i - r_i, \quad \text{where}
\]

\[
R_i = q_i \left[ G_{-i}(r_i) N_1 + (1 - G_{-i}(r_i)) N_2 \right],
\]

and $G_{-i}(r_i) = \text{prob}(r_{-i} < r_i)$. Analogous to the tie breaking rule under the quality-based ranking, in case both developers submit the same bid, developer $H$ will be ranked first. The timing is the same as before, but with an additional choice regarding $r$:

1. **Stage** The app store announces the revenue share $s$.

2. **Stage** The app developers $H$ and $L$ simultaneously choose their qualities $q_L$ and $q_H$, as well as their payments $r_L$ and $r_H$ and the app store ranks the developer with the highest payment first.

3. **Stage** Users choose which app to download, observe the downloaded app’s quality and use it accordingly.

### 4.2 Equilibrium Derivation

We again use backwards induction to solve the game.

**Stage 3:** Stage 3 corresponds to Stage 3 in Section 3.2.

**Stage 2:** This stage consists of two sets of decisions. On the one hand, developers choose their bids in order to compete for the first ranking position by means of an all-pay auction.
On the other hand, developers choose their app qualities. Therefore, under a sponsored ranking regime, the competition for the top position is decoupled from quality setting.

As shown in Riley (1988), there does not exist a pure strategy equilibrium in an all-pay auction. To determine the bounds on $r$, consider that the developers can simply not submit a payment, i.e., $r = 0$. This is the outside option of developer $L$. Under a mixed strategy equilibrium any payment must yield a profit that is at least as high as under the outside option. Assume for a moment that the less efficient developer would win the first position with certainty. Then, let $\bar{r}$ denote the payment for which developer $L$ yields the same profit in the top position than under the outside option. Clearly, any bid higher than $\bar{r}$ would not be reasonable, also not for the $H$ developer, who could just outbid $L$ by also submitting $\bar{r}$.

All bids for the top position will therefore be bound between $\underline{r}$ and $\bar{r}$ and are characterized as follows:

**Lemma 2** (Equilibrium Payment under Sponsored Ranking). Under a sponsored ranking, there exists a unique mixed strategy equilibrium, where app developers $L$ and $H$ choose an expected bid of

$$E[r_L] = \int_{\underline{r}}^{\bar{r}} g_L(r_L) r_L \, dr_L,$$

$$E[r_H] = \int_{\underline{r}}^{\bar{r}} g_H(r_H) r_H \, dr_H$$

in order to achieve the first ranking position. Thereby, developer $L$ ($H$) chooses the minimum bid $\underline{r} = 0$ with probability $G_L(\underline{r}) > 0$ ($G_H(\underline{r}) = 0$), and otherwise chooses a bid between $\underline{r}$ and $\bar{r}$ according to the cumulative distribution function $G_L$ ($G_H$) with corresponding density function $g_L$ ($g_H$). The cumulative distribution functions as well as the minimum and
maximum bids are given by:

\[ G_L = \frac{r_L + (1 - s)(N_1 - N_2) q_H - (N_1 - N_2) q_L}{(1 - s) q_H (N_1 - N_2)} \]

\[ G_H = \frac{r_H}{(1 - s) q_L (N_1 - N_2)} \]

\[ r = 0 \]

\[ r = (1 - s)(N_1 - N_2) q_L. \]

The complete proof is again relegated to the Appendix. The equilibrium bids are exemplified in Figure 2. Notice that the bids of the H developer first-order stochastically dominate the bids of the L developer.

![Figure 2: Distribution of \( r_L \) and \( r_H \) in the mixed strategy equilibrium under sponsored ranking. Numerical example derived for \( s = 0.5, n_1 = 0.75, k = 0.75. \)](image)

Taking into account that the expected profit for each choice of \( r \) equals the outside profit, the optimal quality choice can be readily determined by maximizing the outside profit with respect to quality. This immediately yields the following result.

**Lemma 3** (Equilibrium Quality Choice under Sponsored Ranking). Under a sponsored ranking, there exists a unique pure strategy equilibrium, where app developers L and H choose quality levels of

\[ q_L = \frac{(1 - s) N_2}{2}, \]

\[ q_H = \frac{(1 - s) N_1}{2 k}. \]
Stage 1: Finally, in the first stage, the app store determines the revenue share, $s$, so that $s = \arg \max_s \Gamma$. This yields:

$$s = \frac{2 (N_1)^3 - (k N_2)^2 (N_1 - N_2) - k N_1 N_2 (3 N_1 - 5 N_2)}{4 (N_1)^3 - 4 k N_1 N_2 (N_1 - 2 N_2)}.$$

5 Comparison of App Qualities under Quality-Based and Sponsored Ranking

We now investigate the impact of an app store’s ranking policy on the quality of apps offered by the developers. To this end, it is instructive to compare the expected quality levels as detailed in Lemmas 1 and 3. In the following, we will highlight three effects that explain why and how app qualities may differ across the two regimes. Where necessary, we will differentiate between the variables that relate to the quality-based ranking regime and the variables that relate to the sponsored ranking regime by superscript $q$ and superscript $s$, respectively.

**Competition Effect:** First, note that the qualities chosen under sponsored ranking are identical to those chosen in the pure strategy equilibrium under quality-based ranking (i.e., for $k \leq \bar{k}$), provided the app store chooses the same revenue share, $s = s^q = s^s$, and $\gamma = 1$. That is, whenever the $L$ developer chooses not to engage in the quality-based competition for the first position, both rankings will yield the same app qualities, everything else being equal.

However, when app developers do compete for the first position under a quality-based ranking (i.e., $k > \bar{k}$), then app qualities will be higher than under a sponsored ranking, everything else being equal. To see this, notice first that for $k > \bar{k}$ it holds that $q_L < q_H < q_L = q_H$. Second, recall that under a quality-based ranking the $L$ developer chooses the minimum quality level, $q_L$, with probability $F_L(q_H)$, but with probability
It chooses a higher quality between $q_H$ and $\bar{q}_L$. Third, see from Lemmas 1 and 3 that $q_L$ coincides with the quality level that developer $L$ would (always) choose under a sponsored ranking, everything else being equal. Consequently, under a quality-based ranking, $L$ never chooses a quality level that is lower, but sometimes chooses a quality level that is higher than under a sponsored ranking regime. In other words, when developers are not too different in terms of efficiency (i.e., $k > \tilde{k}$), the expected quality of $L$ is higher under a quality-based ranking, everything else being equal.

In similar vein, see from Figure 3 that for $k > \tilde{k}$ the expected quality of developer $H$ is generally higher under a quality-based ranking than under a sponsored ranking, everything else being equal.

The intuition for this finding is as follows. While app developers compete for the first position under both ranking regimes, they do so with different strategic instruments. Under a sponsored ranking, developers compete with their bids, but not with the app qualities. In comparison, under a quality-based ranking, developers compete directly in app quality, which gives them an additional incentive to set higher quality levels. This is what we denote as the competition effect. In reverse, this means that when the quality-based competition breaks down, then the competition effect vanishes and developers will choose the same quality levels under both regimes.

The competition effect is illustrated in Figure 3. Holding the revenue share fixed across regimes ($s = \frac{1}{2}$) and absent any uncertainty in determining app quality ($\gamma = 1$), the figure depicts the app qualities for all feasible values of the efficiency advantage $k$. Whereas the expected quality levels coincide for $k \leq \tilde{k}$, because the $L$ developer does not compete anymore for the first position, for $k > \tilde{k}$ each developer will choose a higher quality under a quality-based ranking. In fact, notice that when developers are of very similar efficiency ($k \to 1$), then both compete very fiercely for the top position, such that the expected quality of the $H$ and $L$ approach each other, and both developers offer a higher quality than the $H$ developer under a sponsored ranking. To see that this results holds generally, notice from
Lemma 1 that for $k = 1$ it holds that $q_{L} = q_{H}$ and $F_{L} = F_{H}$. Hence, both the respective minimum quality levels, as well as the distribution of the quality choices of the $L$ and $H$ developer approach each other as $k \to 1$. Eventually, both developers choose exactly the same mixed strategy when they are of equal efficiency. Under a sponsored ranking there is no competition in the quality-setting stage, and therefore, even when $k = 1$, developers choose distinct quality levels.

An important corollary that can be gained from this latter insight is that even when app developers are of identical efficiency ($k = 1$), a sponsored ranking amplifies the small competitive disadvantage of the $L$ developer (i.e., that $H$ is chosen as the top ranked developer in case both submit identical bids) to yield very different expected quality levels (and thus profits) for the $L$ and $H$ developer, respectively. Under a quality-based ranking this is not the case, such that, when developers are of identical efficiency, they will also set identical qualities and make identical profits. These results are summarized in the following proposition.

**Proposition 1 (Competition Effect).** The quality competition effect denotes an increase in app quality for each developer when the ranking position is determined (at least partially) by app quality. This effect can only be present under a quality-based ranking and becomes stronger, as app developers become more similar in their ability to produce quality apps, but it is absent when app developers are very different in their ability to produce quality apps ($k \leq \tilde{k}$).

**Scale and Precision Effect:** Due to the economies of scale that stem from the fixed investment costs in quality, the app developer that expects a larger user base is more likely to invest in quality. This is what we denote as the scale effect. The scale effect comes in two flavors. First, even if an app developer would not compete for the first position, it faces a minimum app demand $N_{2}$. The larger $N_{2}$, the higher are the investments in quality, everything else being equal. This is what constitutes a minimum-scale effect that ensures a
Due to the developers’ competition in app quality under a quality-based ranking, expected quality levels are higher for \( k > \tilde{k} \) under a quality-based ranking, everything else being equal. Numerical example derived for \( s = 0.5, n_1 = 0.75, \gamma = 1 \).

certain minimum app quality. Second, when developers compete for the top position, they do so, because this secures them additional downloads of \( N_1 - N_2 \). This is what constitutes the additional-scale effect. Evidently, in our set-up with \( \partial N_1 / \partial n_1 > 0 \) and \( \partial N_2 / \partial n_1 < 0 \), an increase in \( n_1 \) will lead to a reduction in \( N_2 \), i.e., to a decrease in the minimum-scale effect, and to an increase in \( N_1 \), i.e., to an increase in the additional-scale effect. For the \( H \) developer, who wins the top position more often, the increase in additional-scale effect dominates, such that it increases its expected quality with an increase in \( n_1 \). For the \( L \)-developer, who wins the second position more often, the minimum-scale effect dominates, such that it decreases its expected quality as \( n_1 \) increases. This trade-off between the minimum-scale and additional-scale effect is demonstrated in Figure 4a. Therein, note that the scale effect acts similar under both regimes and does not render any differences per se. The fact that the expected qualities are higher under a quality-based ranking is due to the competition effect, not due to the scale effect. This means that, for a qualitative comparison of the two ranking regimes, the magnitude of \( n_1 \) is not important.

However, it is important for the comparison of the two regimes that under a sponsored ranking the app store can perfectly identify the highest bidding developer, whereas under
Figure 4: Precision and Scale Effects. (a) Scale effect: When $N_1$ increases (i.e. $N_2$ decreases), $H$ ($L$) developers expect a larger (lower) number of downloads, inducing them to invest more (less) in quality. (b) Precision effect: Due to the scale effect, a decrease in the app store’s precision to detect quality, $\gamma$, yields a decrease (increase) in the $H$ ($L$) developer’s expected quality, but only under a quality-based ranking. Numerical example derived for $s = 0.5$, $k = 0.75$, $n_1 = 0.75$, $\gamma = 1$.

In reverse, under a sponsored ranking the developer with the lower bid expects a scale of $N_2^s = (1 - n_1)$, but under a quality-based ranking the developer of the app with the lower quality expects a scale of $N_2^q = (1 - \gamma)n_1 + \gamma(1 - n_1) \geq N_2^s$. Consequently, for $\gamma < 1$, everything else being equal, the scale effect is the reason why it is relatively less attractive to invest in a high quality app under a quality-based ranking, whereas it is relatively more attractive to invest in the low quality app. This is what we denote as the precision effect. Of course, the precision effect follows immediately from the scale effect, but it is only present under a quality-based ranking. This is exemplified in Figure 4b and summarized in the following proposition.
Proposition 2 (Scale and Precision Effect). When the top ranking position becomes more important to attain downloads ($N_1$ increases and $N_2$ decreases), the high-efficiency developer increases and the low-efficiency developer decreases its expected quality under each ranking regime, everything else being equal (scale effect). An increase in the app store’s precision to determine app quality ($\gamma$ increases) acts like a scale effect, but only for the quality-based ranking (precision effect).

Revenue-Share Effect: Finally, notice from Lemmas 1 and 3 that the quality levels under both regimes are immediately affected by the revenue share $s$. The larger the share $s$, the lower the expected quality level. Whereas the equilibrium revenue share is constant at $s^q = 1/2$ under a quality-based ranking, it is strictly smaller under a sponsored ranking for $k \in (0, 1]$. This is because under sponsored ranking the app store can, in addition to the revenue share, also expropriate revenues via the ranking payment. Figure 5a exemplifies that, as $k$ increases, $s$ becomes smaller under a sponsored ranking. For $k = 0$ the revenue share coincides under both regimes. Thus, by the revenue-share effect, app developers have a stronger incentive to invest in app quality under a sponsored ranking, everything else being equal.

Proposition 3 (Revenue-Share Effect). The revenue-share effect denotes to an increase in app quality when the app store demands a lower revenue share, $s$. For all feasible parameter values, the app store will choose a lower revenue share under sponsored ranking in equilibrium.

Interaction of Effects: First, let us abstract from the precision effect by assuming $\gamma = 1$ and focus on the interaction between the competition and the revenue-share effects. Both act in similar fashion on both types of developers. Whereas the competition effect is stronger under a quality-based ranking, the revenue-share effect is stronger under a sponsored ranking. Thus, it will depend on the relative strength of these two effects which ranking regime will overall lead to higher expected qualities by each type of developer. Recall that in the previous
figures we have deliberately neglected the revenue-share effect by holding "everything else equal" and fixing a certain level of $s$ across all regimes. Instead, in Figure 5b we apply the actual, regime specific equilibrium value of $s$ at $\gamma = 1$. It can be seen that for low levels of $k$, higher qualities are expected under a sponsored ranking, because the competition effect is relatively weak (in fact zero for $k \leq \tilde{k}$) and thus the revenue-share effect dominates. On the contrary, if $k$ is close to one, the competition effect dominates the revenue-share effect and higher qualities are expected under a quality-based ranking. At $\gamma = 1$, this applies to the quality levels of both the $H$ and the $L$ developer respectively, although at different critical values of $k$. Generally, the competition effect will dominate the revenue-share effect earlier for the $L$ developer, yielding a lower critical threshold for $k$ at which the $L$ developer produces a higher quality under a quality-based ranking compared to a sponsored ranking.

![Figure 5: Revenue-Share Effect](image_url)

(a) Equilibrium Revenue-Share $s$

(b) Revenue-Share Effect vs. Competition Effect

However, at levels of $\gamma < 1$, the precision effect will induce the $H$ developer to decrease, and the $L$ developer to increase its quality under a quality-based ranking, relative to the
level that would have been set at $\gamma = 1$ (see Proposition 2). The interaction between $k$, which governs the trade-off between the competition and revenue-share effects, and $\gamma$, which governs the trade-off between the minimum-scale and additional-scale effects under a quality-based ranking, is demonstrated in Figure 6 for all feasible values of $k$ and $\gamma$. Note that the figure provides a complete characterization of the comparison of app qualities between the two regimes, because, as noted earlier, the only remaining parameter $n_1$ does not have a qualitative effect on outcomes.

![Figure 6: Comparison of App Qualities between Quality-Based and Sponsored Ranking. Parameter regions of $k$ and $\gamma$ where a quality-based ranking yields a higher expected quality, $q_i^q > q_i^s$ than sponsored ranking. In the cross-hatched area, a quality-based ranking yields higher expected qualities for both developers. In the remaining diagonally-shaded area, a quality-based ranking yields a higher expected quality for the $L$ developer, but not for the $H$ developer. In the non-shaded area, a sponsored ranking yields a higher expected quality for both developers. Numerical example derived for $n_1 = 0.75$ and equilibrium values for $s$.](image)

It can be seen that quality-based ranking dominates sponsored ranking (in terms of the expected qualities set by the $H$ and $L$ developers, respectively) only when developers are relatively similar ($k$ is large, i.e., the competition effect is large), and, at the same time, the app store has high precision in determining app quality ($\gamma$ is large, i.e., the scale effect of the $H$ developer is not diminished). However, for a wide range of parameter values (except for low values of $k$ combined with high values of $\gamma$), the $L$ developer will offer a higher quality
under a quality-based ranking. This is the result of the combined effect of the increased scale effect of the $L$ developer due to low precision in determining app quality under a quality-based ranking, and the relatively strong competition effect, which is absent under a sponsored ranking.

Taken together, these insights already hint to the fact that a sponsored ranking amplifies even small existing differences in app developers’ efficiency to generate a larger spread of expected quality levels between the $H$ and $L$ developer, $\Delta = q_H - q_L$, than a quality-based ranking. In the Appendix we show that this is true for all parameter regions.

The results from this section are summarized in the following proposition.

**Proposition 4** (Comparison of App Qualities). Generally it is ambiguous which ranking regime leads to a higher app quality. A quality-based ranking unambiguously leads to higher app qualities than a sponsored ranking only if developers are similar in efficiency and the app store’s ability to accurately assess app quality is high. However, under a quality-based ranking the expected quality spread between the low- and high-efficiency developer is always lower than under a sponsored ranking.

## 6 Biased Ranking

### 6.1 The model

So far we have considered unbiased ranking regimes, i.e., regimes that rank the app developers according to some predetermined and (possibly imperfectly) observable measure (either $q$ or $r$). We now study the case of a biased ranking regime, where the first ranking position is solely based on the app developers’ identity ($H$ or $L$). That is, we now consider the limiting case where either the $H$ or the $L$ developer is always promoted by the app store, and this is common knowledge. This means that the promoted developer always attains $N_1 = n_1$ and the other developer always attains $N_2 = (1 - n_1)$ downloads. It is useful to study this
case, because it eliminates any type of competition for the first rank between the two app developers. The timing is then as follows:

1. **Stage** The app store announces its revenue share $s$.

2. **Stage** The app developers $H$ and $L$ simultaneously choose their app qualities $q_H$ and $q_L$, and the app store grants the first position of the ranking always to the promoted app developer (either $H$ or $L$).

3. **Stage** Users choose which app to download, observe the downloaded app’s quality and use it accordingly.

### 6.2 Equilibrium Derivation

**App store promotes the high efficiency developer:** First, we consider the case where the app store is biased towards the $H$ developer. We again use backwards induction to derive the equilibrium. The analysis of Stage 3 is identical to the analysis of Stage 3 in Section 3.2. In Stage 2 the developers know with certainty that the $H$ developer will always obtain the first, and that the $L$-developer will always achieve the second ranking position. Thus, there is no competition for the first rank, neither in qualities nor in bids. Therefore, the optimal qualities are readily given by profit maximization:

\[
q_{b,H}^L = \arg \max_{q_L} \Pi_{b,H}^L = \frac{(1 - s) N_2}{2}, \tag{8}
\]

\[
q_{b,H}^H = \arg \max_{q_H} \Pi_{b,H}^H = \frac{(1 - s) N_1}{2 k}. \tag{9}
\]

In Stage 3, anticipating its impact on the quality $q_L$, the integrated app store maximizes its profit by setting the revenue share to $s = \arg \max_s \Gamma = \frac{1}{2}$.

**App store promotes the low efficiency developer:** The same logic can be applied when the app store’s ranking is biased towards the $L$ developer. Accordingly, in Stage 2 we
obtain

\[ q_{b,L}^{b,L} = \arg \max_{q_H} \Pi_{b,L}^{b,L} = \frac{(1 - s) N_2}{2k}, \quad (10) \]

\[ q_{b,L}^{b,L} = \arg \max_{q_L} \Pi_{b,L}^{b,L} = \frac{(1 - s) N_1}{2}, \quad (11) \]

and in Stage 3 the app store would again set a revenue share of \( s^{*} = \arg \max_{s} \Gamma = \frac{1}{2} \).

**Lemma 4** (Equilibrium Quality Choice under a Biased Ranking). Under a biased ranking, in which one of the two app developers is always awarded the top position in the ranking, there exists a unique pure strategy equilibrium. If the more efficient developer is promoted, then it chooses a quality level of \( q_{b,H}^{b,H} = \frac{(1 - s) N_1}{2k} \), and the less efficient app developer a quality level of \( q_{b,L}^{b,L} = \frac{(1 - s) N_2}{2} \). If the less efficient developer is promoted, then it chooses a quality level of \( q_{b,L}^{b,L} = \frac{(1 - s) N_1}{2} \), and the more efficient app developer a quality level of \( q_{b,L}^{b,H} = \frac{(1 - s) N_2}{2k} \).

### 6.3 Comparison of App Qualities under Biased Ranking

In order to understand how a biased ranking impacts app quality, it is once again instructive to consider competition, scale, precision and revenue effects in the two cases.

**App store promotes the high efficiency developer:** First, it is evident that the competition effect is absent under a biased ranking. In this regard, it is similar to a sponsored ranking and, everything else being equal, quality-based ranking should yield a higher app quality than a biased ranking (or sponsored ranking).

Second, notice that in case the \( H \) developer is promoted, the app qualities are exactly the same as under a sponsored ranking, provided the revenue-share \( s \) is the same, because in both rankings the developers do not compete in qualities. However, the revenue-share effect is stronger under a sponsored ranking compared to a biased ranking, because under a sponsored ranking the app store will set a lower revenue share of \( s^{*} < \frac{1}{2} \). Consequently,
even when the app store is biased towards the more efficient app developer, the app qualities of each developer will be lower than under a sponsored ranking.

Third, under a biased ranking where $H$ is promoted, the scale effect acts similar as in the other two unbiased ranking regimes. However, similar as under a sponsored ranking, there is no uncertainty in determining who should be awarded the top ranking position. Thus, in comparison to a quality-based ranking, for low values of $\gamma$, everything else being equal, a biased ranking (like a sponsored ranking) should yield a higher expected quality of the high quality app, and a lower expected quality of the low quality app. Together with the previous point, this also means that for $\gamma = 1$ a biased ranking always yields lower qualities than a quality-based ranking, for all values of $k$. This is visualized in Figure 7a.

**App store promotes the low efficiency developer:** Generally the same reasoning as above applies with respect to the competition, revenue-share and precision effect in case the $L$ developer is promoted under a biased ranking. However, an additional distortion arises due to the scale effect. In an unbiased ranking, or in a ranking that is biased towards $H$, the $H$ developer will always be awarded the top position more often than the $L$ developer.\(^5\) This means that the $H$ developer enjoys a larger scale effect in these cases, but a lower scale effect in a biased ranking where $L$ is promoted. This means that, in a biased ranking where $L$ is promoted, depending on the size of the scale advantage ($n_1$) relative to the efficiency disadvantage ($k$), the $L$ developer may even offer a higher quality than the $H$ developer. However, since the $H$ developer is more efficient, given the same scale effect, it would offer a higher app quality than the $L$ developer. Therefore, $L$’s quality is always below the top quality (provided by the $H$ developer) that would have emerged under a sponsored ranking, or a biased ranking where $H$ is promoted. Of course, in reverse this means that in a biased ranking where $L$ is promoted, the $H$ developer’s quality will always be higher than the lowest quality (provided by the $L$ developer) under a sponsored ranking. But it is also clear that the average app quality is lower in a biased ranking that promotes $L$, than in a biased ranking

\(^5\)Recall that $F_H$ and $G_H$ stochastically dominate $F_L$ and $G_L$, respectively.
that promotes $H$, as long as $k < 1$. Finally, notice again that a biased ranking would yield worse quality levels than a sponsored ranking even when $k = 1$ due to the lower revenue-share effect. The quality levels under a biased ranking where $L$ is promoted are exemplified in Figure 7b.

![Figure 7](image)

(a) $H$ Developer is Promoted
(b) $L$ Developer is Promoted

Figure 7: Expected Quality Levels Under Biased Ranking. (a) When the $H$ developer is promoted, $H$’s and $L$’s quality are lower than under sponsored ranking due to the lower revenue-share effect. (b) When the $L$ developer is promoted, the top quality is lower than if $H$ is promoted; and $H$’s quality is significantly lower than under a sponsored ranking due to the lower scale and revenue-share effects. Numerical example derived for $n_1 = 0.75$, $\gamma = 1$ and equilibrium values for $s$.

**Proposition 5** (Biased Ranking). If in a biased ranking, where the more efficient app developer is always awarded the top position, the app developers’ qualities are strictly lower than their respective quality levels under a sponsored ranking. If the less efficient app developer is always awarded the top ranking position, it may offer a higher quality than the $H$ developer, but the top quality and the average app quality are (weakly) lower than under a biased ranking where the $H$ developer is promoted.

---

6This is easy to verify with the equilibrium app qualities given in Lemma 4.
7 Vertically Integrated App Developer and Biased Ranking

7.1 The Model

We now consider the special case of a biased ranking where the app store is vertically integrated with one of the developers and will always promote its own, vertically integrated app to the top position. Again, we consider both cases, i.e., the integrated app developer being of higher efficiency \(H\) developer) or lower efficiency \(L\) developer) than the competing app developer. The profit function of the vertically integrated app store and app developer is

\[
\Gamma = R_\Gamma - C_\Gamma + s R_{-\Gamma},
\]

(12)

where \(R_\Gamma\) is the revenue, and \(C_\Gamma\) is the cost function of the integrated app developer/store, and \(s R_{-\Gamma}\) the revenue share that is collected from the non-integrated developer. Note that, due to vertical integration, \(R_\Gamma\) is independent of \(s\). Also note that, from the perspective of the integrated app developer, this is equivalent to the case where it is offered a revenue share of \(s = 0\) by the app store. The timing is now as follows:

1. **Stage** The app store announces its revenue share \(s\) for the non-integrated developer.

2. **Stage** The app store and the non-integrated app developer set their app qualities \(q_\Gamma\) and \(q_{-\Gamma}\), and the app store always grants the first position of the ranking to its own app.

3. **Stage** Users choose which app to download, observe the downloaded app’s quality and use it accordingly.
7.2 Equilibrium Derivation

**Integrated app developer is of higher efficiency:** We begin again by studying the case where the integrated app developer is the $H$ developer, i.e., of higher efficiency than the non-integrated rival. The analysis proceeds as in Section 6.2. Consequently, optimal qualities in Stage 2 are readily given by profit maximization:

$$q_{v,H}^v = \arg \max_{q_L} \Pi_{v,H}^v = \frac{(1-s)N_2}{2},$$

$$q_{v,H}^\Gamma = \arg \max_{q_{\Gamma}} \Pi_{v,H}^\Gamma = \frac{N_1}{2k}.$$  

(13)  

In Stage 1, anticipating its impact on the quality $q_{L}$, the integrated app store maximizes (12) by setting the revenue share to $s = \arg \max_{s} \Gamma = 1/2$ for the non-integrated app developer.

**Integrated app developer is of lower efficiency:** When the integrated app developer is the $L$ developer, and the non-integrated developer is the $H$ developer, i.e., the integrated app developer is of lower efficiency, optimal app qualities are given by

$$q_{v,L}^v,H = \arg \max_{q_H} \Pi_{v,L}^v,H = \frac{(1-s)N_2}{2k},$$

$$q_{v,L}^\Gamma = \arg \max_{q_{\Gamma}} \Pi_{v,L}^\Gamma = \frac{N_1}{2}.$$  

(15)  

The optimal revenue-share is again set to $s = \arg \max_{s} \Gamma = 1/2$ for the non-integrated app developer. Taken together, this yields the following lemma.

**Lemma 5** (Equilibrium Quality Choice under Vertical Integration). Under vertical integration, where the vertically integrated developer is always awarded the top position in the ranking, there exists a unique pure strategy equilibrium. If the integrated developer is of higher efficiency, then it chooses a quality level of $q_{v,H}^v = \frac{N_1}{2k}$, and the non-integrated app developer a quality level of $q_{L}^v,H = \frac{(1-s)N_2}{2}$. If the integrated developer is of lower efficiency,
then it chooses a quality level of $q_{v,L}^v = \frac{N_2}{2}$, and the non-integrated app developer a quality level of $q_{H}^{v,L} = \frac{(1-s)N_2}{2k}$.

7.3 Comparison of App Quality under Vertical Integration

**Integrated app is of higher efficiency:** It is evident that the same reasoning as in a biased ranking where $H$ is promoted also applies in this case. However, due to vertical integration the revenue effect now acts differently on the two app developers. Whereas the app store demands a revenue share of $s = \frac{1}{2}$ from the non-integrated developer (as under a biased ranking), it demands a revenue share of $s = 0$ from the integrated developer. Consequently, the revenue-share effect acts on the non-integrated $L$ developer as under a biased or quality-based ranking, i.e., it is weaker than under a sponsored ranking. However, the absence of a revenue share provides the integrated $H$ developer with strong incentives to invest in quality under vertical integration, much stronger than under either quality-based, sponsored or biased ranking without vertical integration. Consequently, by the revenue-share effect the integrated $H$-developer is expected to set a higher app quality than under sponsored ranking and quality-based ranking, everything else being equal.

Taken together, by the competition and scale effects, vertical integration of the $H$ developer with the app store will yield similar outcomes as under a sponsored ranking. However, by the additional revenue-share effect, the quality of the integrated $H$ developer is expected to be higher than under a sponsored ranking, whereas the quality of the $L$ developer is expected to be slightly lower. Consequently, when the $H$ developer is vertically integrated, the expected quality spread between the high- and low-efficiency developer is going to be even higher than under a sponsored ranking. This is visualized in Figure 8a, where the app qualities are shown for all feasible values of $k$. Notice that the revenue-share effect of the vertically integrated $H$-developer can even be so large that vertical integration yields a higher expected app quality of the $H$ developer than any other regime. In contrast to the corresponding biased ranking without vertical integration, with vertical integration the
average expected app quality can therefore be higher than under a sponsored ranking. This is demonstrated in Figure 9.

![Graphs showing expected quality levels under vertical integration]

(a) $H$ developer is vertically integrated
(b) $L$ developer is vertically integrated

Figure 8: Expected Quality Levels Under Vertical Integration. (a) When the $H$ developer is vertically integrated, $H$’s quality is higher and $L$’s quality is lower than under sponsored ranking due to the revenue-share effect. (b) When the $L$ developer is vertically integrated, due to the increased revenue-share and scale effect, $L$’s quality can be higher than $H$’s quality under sponsored ranking, but only if $L$’s efficiency disadvantage is not too large. Numerical example derived for $n_1 = 0.75$, $\gamma = 1$ and equilibrium values for $s$.

**Integrated app is of lower efficiency:** Generally the same trade-offs along the competition, scale, precision and revenue-share effects as in a biased ranking where $L$ is promoted also exist when the $L$ developer is vertically integrated. However, here again the revenue-share effect acts differently on each developer. In this case, the $L$ developer’s quality is expected to be significantly higher than in the corresponding biased ranking without vertical integration, because it experiences a much larger revenue-share effect. Nevertheless, the average app quality is still lower than in the case where $H$ is vertically integrated, because the less efficient developer is promoted to the top position. However, due to the large revenue-share effect for the vertically integrated $L$ developer, the quality of the high quality app (provided by $L$ under vertical integration) may still be higher than the quality of the high quality app (provided by $H$) under sponsored ranking or quality-based ranking. Figure
8b exemplifies this. Consequently, in contrast to the corresponding biased ranking without vertical integration, with vertical integration the average expected app quality can yet be higher than under an unbiased ranking (see Figure 9).

**Proposition 6 (Vertical Integration).** If the more efficient app developer is vertically integrated and always awarded the top ranking position, its app quality is strictly higher, and the app quality of the less efficient developer is weakly lower than the respective developer’s expected app quality under a sponsored ranking regime. If the less efficient app developer is vertically integrated and always awarded the top ranking position, its app quality is strictly higher, and the app quality of the more efficient developer is weakly lower than the respective developer’s expected app quality under a sponsored ranking regime. In both cases the top app quality and the average expected app quality can be higher than under a quality-based or sponsored ranking.

![Comparison of Average Expected Quality](image)

Figure 9: *Comparison of Average Expected Quality.* Numerical example derived for $n_1 = 0.75$, $\gamma = 1$ and equilibrium values for $s$. 
8 Welfare Analysis

Finally, we compare the sponsored ranking and quality-based ranking regime also from a welfare perspective. We measure welfare by i) consumer surplus, ii) app store surplus, iii) app developers’ surplus, and iv) total surplus. From an antitrust perspective, consumer surplus will be of paramount importance, especially in the US. However, it can be argued that app developers’ surplus is more important with respect to the long-run economic well-being of the app economy. The app store’s surplus is important for determining which ranking regime would be preferred by the app store, and, when contrasted with consumer surplus or total surplus, market failures can be identified.

**Consumer Surplus:** Since users can download apps for free, a higher average app quality will immediately translate into a higher consumer welfare. Therefore, we make the assumption that the surplus of a representative consumer is simply given by \( U = q_i \), where \( q_i \) denotes the quality of the app that the consumer has downloaded.\(^7\) Consequently, since the total mass of consumers is \( N_1 + N_2 = 1 \), consumer surplus is given by the average weighted expected quality as follows

\[
CS = \hat{q} = N_1 \hat{q}_1 + N_2 \hat{q}_2,
\]

where \( \hat{q}_i \) denotes the average expected quality of the app in the \( i \)th position. In a quality-based ranking, we have

\[
\hat{q}_1^q = F_H(q_H) F_L(q_H) q_H + \left( \int_{q_H}^{q_L} (F_L f_H q_H) dq_H \right) + \left( \int_{q_H}^{q_L} (F_H f_L q_L) dq_L \right)
\]

\[
\hat{q}_2^q = F_H(q_H) \left( 1 - F_L(q_H) \right) q_H + \left( \int_{q_H}^{q_L} ((1 - F_L) f_H q_H) dq_H \right)
\]

\[
+ F_L(q_H) q_L + \left( \int_{q_H}^{q_L} ((1 - F_H) f_L q_L) dq_L \right),
\]

\(^7\)This is to say that we assume a standard quasi-linear utility function \( U = vq - p \), where \( p = 0 \), since apps are provided for free, and where we normalize \( v \) to be one, without loss of generality.
with $N_1^q = \gamma n_1 + (1 - \gamma)(1 - n_1)$ and $N_2^q = (1 - \gamma) n_1 + \gamma (1 - n_1)$.

In a sponsored ranking, we have

$$\hat{q}_1^s = \left( \int_0^\tau (G_L g_H) dr_H \right) q_H^s + \left( \int_0^\tau (G_H g_L) dr_L \right) q_L^s$$

$$\hat{q}_2^s = \left( \int_0^\tau ((1 - G_L) g_H) dr_H \right) q_H^s + \left( \int_0^\tau ((1 - G_H) g_L) dr_L \right) q_L^s,$$

with $N_1^s = n_1$ and $N_2^s = (1 - n_1)$.

Figure 10 compares the consumer surplus between a quality-based ranking and sponsored ranking over the full range of feasible parameter values for $\gamma$ and $k$. It can be seen that quality-based ranking yields a higher consumer surplus only if the app store can assess app quality relatively well ($\gamma$ is sufficiently large), and app developers differ not too much in efficiency ($k$ is sufficiently high). In the light of our previous analysis, this means that a quality-based ranking leads to a higher consumer surplus only if the competition effect and the precision effect are relatively strong.

**App Developers’ Surplus:** The app developers’ surplus is readily given by $DS = \Pi_H + \Pi_L$, where $\Pi_i$ is given in equations (3) and (6) for the quality-based and the sponsored ranking regime, respectively. Figure 10 also compares the developer surplus across the two regimes. It can be seen that a quality-based ranking yields a higher surplus for developers only if the precision effect is strong ($\gamma$ is large), and the competition effect is weak, or even absent ($k$ is small). This is because, when the competition effect is strong, app developers invest much more in quality, which is good for consumers, but bad for developers’ surplus, as it incurs large fixed costs. Consequently, app developers prefer a quality-based ranking over a sponsored ranking in about the same parameter region where a quality-based ranking yields an unambiguously lower app quality (see Figure 6).

---

8Recall that the remaining parameter $n_1$ has no qualitative effect on market outcomes, and, thus, on the welfare comparison.
Figure 10: *Comparison of Consumer and Developers’ Surplus.* Comparison of consumer surplus and developers’ surplus between the quality-based and the sponsored ranking over the full range of parameter values for $k$ and $\gamma$. In the diagonally-shaded area, quality-based ranking yields higher expected surplus for consumers. In the horizontally-shaded area, quality-based ranking yields a higher surplus for the developers. In the non-shaded area, sponsored ranking yields a higher expected surplus for consumers and developers. Numerical example derived for $n_1 = 0.75$ and equilibrium values for $s$.

However, by just considering the combined developer surplus, we would mask the fact that the combined surplus is indeed mostly driven by the $H$ developer. Figure 11 highlights that the $L$ developer would prefer a quality-based ranking for a much larger range of parameters than the $H$ developer. This is mostly due to the (lack of) precision effect, which leaves the $L$ developer relatively better off when $\gamma$ is low. In fact, notice that for $\gamma = 1$ the $L$ developer would never prefer a quality-based ranking over a sponsored ranking. In this context, recall also that the sponsored ranking amplifies the quality difference between the $H$ and the $L$ developer (see Proposition 4.)

**App Store Surplus and Total Surplus:** The app store surplus is given by $\Gamma$ in equation (4). It turns out that, in equilibrium, the app store makes higher revenues under the sponsored ranking regime exactly in the same parameter regions where the consumers receive a higher surplus under sponsored ranking (compare Figure 10). Thus, if the app store were to
Figure 11: Comparison of Developers’ Profits. Comparison of developer $H$’s and developer $L$’s preferred ranking regime for all parameter values of $k$ and $\gamma$. In the diagonally-shaded area, quality-based ranking yields higher expected profit for $L$. The horizontally-shaded area yields higher expected profit for $H$ under quality-based ranking. Numerical example derived for $n_1 = 0.75$ and equilibrium values for $s$.

choose a ranking regime, say in a Stage 0 of an extended game, then it would always choose exactly the regime that would maximize consumer surplus. This is because the app store makes revenues based on the app developers’ expected gross revenues (net of a possible ranking payment), but not based on app developers’ profits (i.e., not including the investment costs). App developer’s gross revenues are maximized when the average weighted expected quality of the app developers is maximized, i.e., when consumers’ surplus is maximized. Therefore, when choosing the ranking regime, the app store’s incentives and the consumers’ incentives are perfectly aligned.

However, the app store’s preferred ranking regime decision ignores the app providers’ investment costs. Therefore, with respect to total welfare, market failures can nevertheless occur. This is exemplified in Figure 12, which highlights that two type of market failures can occur: At high values of $\gamma$ the app store may prefer a sponsored ranking, although a quality-based ranking would maximize total surplus, and at low values of $\gamma$ the app store may prefer a quality-based ranking although a sponsored ranking would maximize total surplus.
Figure 12: Comparison of Total Surplus and App Store Surplus. Comparison of total surplus and app store surplus (as well as consumer surplus) for all parameter values of $k$ and $\gamma$. In the diagonally-shaded area, quality-based ranking yields higher total surplus. In the horizontally-shaded area, a quality-based ranking yields a higher app store surplus (and consumer surplus). In the non-shaded area, sponsored ranking yields higher total surplus and app store surplus (and consumer surplus). Numerical example derived for $n_1 = 0.75$ and equilibrium values for $s$.

**Proposition 7** (Welfare). A quality-based ranking yields a higher consumer or total surplus than sponsored ranking only if the app store can assess app quality relatively accurately ($\gamma$ large) and the app developers are relatively similar in efficiency ($k$ large). The app store would always prefer the ranking regime that maximizes consumer surplus. However, the app store may not prefer the regime that maximizes total surplus.

**9 Conclusions**

**Summary and Theoretical Contribution:** App stores are important gatekeepers of the digital economy, mediating the exchange between app developers and app users. Therefore, changes in the app store’s ranking regime can have far reaching economic consequences. We developed a game theoretic model with a monopoly app store and two app developers, in order to investigate how a sponsored ranking regime affects app quality, profits, and welfare.
in comparison to a quality-based ranking regime. We fully characterized the possible market outcomes and identified three distinct economic incentives (competition effect, scale and precision effect, revenue-share effect) that jointly determine app developers’ incentives to invest in app quality, and hence welfare. We could show that a sponsored ranking regime can lead to higher app quality and welfare, particularly when the app store can only imperfectly observe app quality (low precision effect) or when app developers are quite asymmetric in their ability to produce quality apps (low competition effect). However, we also highlighted that a sponsored ranking tends to favor those app developers that already have an efficiency advantage, and thus, the expected quality spread between the low- and high efficiency developer is always higher under a sponsored ranking.

Finally, we also considered the impact of a biased ranking on app quality. Absent vertical integration, a biased ranking always yields a lower expected quality level than a sponsored ranking and is therefore welfare decreasing. However, if one of the app developers is vertically integrated with the app store, then this provides strong incentives for the integrated developer to invest in quality, which can ultimately yield a higher average app quality than in any other regime. Clearly, our results bear a number of important managerial and policy implications.

**Managerial Implications:** From a managerial perspective, our model highlights that a sponsored ranking may not be bad for developers, especially those who can produce quality apps with high efficiency. Especially in an environment where it is difficult for the app store to assess app quality accurately, or where the app store applies an intransparent editorial policy, the developers may find that the bidding process under a sponsored ranking signals their app quality more effectively than the app store’s noisy (and potentially biased) quality assessment process.

Moreover, a sponsored ranking shields developers from engaging in a costly quality-based competition. Competition in quality is costly for developers, because the cost function is convex and thus, marginal investments in “quality” become increasingly expensive. However,
competition in “money” entails constant marginal costs. Thus, a sponsored ranking can yield higher profits for developers, because it prevents them from excessive investments in quality, above and beyond the quality level that they would offer absent any competition.

However, a sponsored ranking may also pick winners and losers. Those developers that have a competitive disadvantage are likely to be comparably worse off under a sponsored ranking. In particular, we have shown that under a sponsored ranking even a very small difference in the app developers’ inefficiencies will, in contrast to a quality-based ranking, yield a distinct differentiation in app quality, because the developers do not compete in quality. Under a sponsored ranking, the less efficient developers are also comparably worse off, because they do not benefit from errors in the app store’s ranking algorithm anymore, which had secured them additional downloads under quality-based ranking regime.

**Policy Implications:** From a policy perspective, an important observation is that the app store’s and consumers’ preferred ranking regime choice are perfectly aligned. This is because the app store prefers the regime that yields the higher weighted average of the expected app qualities, which is a proxy for the gross revenues that app developers receive, and which therefore corresponds to a higher profit of the app store. Thus, our model provides an example of an online market, where the exercise of market power by a dominant platform does not translate into a loss of consumer welfare. Yet, considering a total welfare standard, market failures may still occur. This supposed shortcoming of the consumer welfare standard, which is adopted, e.g., in US antitrust enforcement, has recently led to a lively debate regarding the appropriate welfare standard for policymakers (see, e.g., Khan, 2017). Our model exemplifies this trade-off between consumer and total welfare in the context of the digital economy.

In Europe policymakers have proposed to impose platform neutrality regulation for dominant online platforms, such as app stores. Extending the legal framework of net neutrality, which currently only applies to Internet Service Providers, would possibly prohibit online
platforms to offer pay-for-prominence regimes, such as a sponsored ranking (cf. Easley et al., 2018). Our results are a critical assessment in this regard. The application of such a rule would crucially depend on the market environment and its intended purpose. In general terms, we identify many instances in which a sponsored ranking regime would yield better results (in terms of quality, profits and welfare) than a quality-based ranking regime. Thus, we find only weak support for strict ex-ante regulation of ‘app store neutrality’. If, however, the purpose of the rule is to ensure competition between app developers, especially in order to protect ‘small’ and less efficient app developers, then a ‘neutrality’ rule may be warranted.

A neutrality rule has also been proposed in an effort to prevent the app store to favor its own, vertically integrated app. We show that vertical integration need not be to the detriment of consumers, because it can boost the quality of the integrated app due to a surge in the revenue-share effect. However, absent vertical integration a biased ranking will lead to lower quality apps. Thus, vertical integration may not be seen as problematic in this environment. But preventing biases, e.g., through demanding more transparency from app stores with regard to the measures that influence a ranking position, can ensure fair and effective competition, which will improve market outcomes.

Limitations and Future Research: In closing, we would also like to mention some limitations of our model that may prompt future research. First, we have highlighted in the introduction that app stores enjoy market power and therefore considered a monopolistic app store. Yet, two dominant app stores exist, and a possible policy agenda may be to introduce more competition among them. Therefore, it would be insightful to see how the competition between app stores affects developers’ quality choice and the app store’s choice of the ranking regime. However, this would require several significant changes to the model setup, such as the introduction of a third developer, which is beyond the scope of this paper.

Second, we have considered app developers that can only invest in quality that increases consumers’ utility. One could also think of investments that decrease consumers’ utility, such
as more intrusive advertising technology or other privacy diminishing measures. A sponsored ranking may increase the number of downloads of such malicious apps, because it would be possible to attain a high ranking position despite being of low quality for users. If this is true and of a dominant concern, this would clearly change our results and policy implications, and may support the application of a neutrality rule after all.

Third, in this paper we have considered the trade-offs between either a pure quality-based, or a pure sponsored ranking regime. In reality, a hybrid sponsored ranking mechanism, which gives bids of app developers with higher quality more weight, everything else being equal, may strike a balance between both regimes. On the one hand, a hybrid sponsored ranking would provide an additional quality-competition boost over a pure sponsored ranking, giving developers more incentive to invest in app quality. On the other hand, it would reduce the revenue-share effect and introduce a noisy quality assessment in comparison to a pure sponsored ranking. Overall, the effect of a hybrid sponsored ranking on app quality will heavily depend on how exactly app quality is factored into the bids. We therefore believe that our analyses of a pure quality-based and a pure sponsored ranking regime provide important bounds for the spectrum of feasible outcomes that may emerge under a hybrid sponsored ranking, and we leave it to future research to explore this further.

References


A Proof of Lemma 1

To characterize these equilibria it is necessary to determine the reasonable quality range for each developer. At first we have to consider that $L$ can choose not to compete in qualities i.e. to serve only $N_2$ users. L’s profit in this situation is

$$\Pi_L = (1 - s)q_L N_2 - q_L^2.$$  \hfill (18)

Maximization with respect to $q_L$ results to

$$q_L = \frac{(s - 2)N_2}{2}.$$  \hfill (19)
By inserting Equation (19) into (18) we obtain an outside profit of

\[ \tilde{\Pi}_L = q_L^2. \]  
(20)

Suppose there is a \( q_L > q_H \), resulting in a profit of \( \hat{\Pi} \) for which \( L \) obtains the top position with probability \( \gamma \). \( q_L \) is a reasonable choice if \( \hat{\Pi}_L \geq \tilde{\Pi} \) holds. We can rearrange this to determine the maximum reasonable quality of developer \( L \) as

\[ q_L = \frac{(1 - s) (N_1 + \sqrt{N_1 - N_2})}{2}. \]  
(21)

This quality level is sufficient to outbid the \( L \) developer. Choosing \( q_H = q_L \) provides it with an outside profit of

\[ \tilde{\Pi}_H = (1 - s) q_L N_1 - C_H(q_L). \]  
(22)

Note that \( H \) chooses \( q_H \geq q_L \) if \( q_L \leq \arg \max_{q_H} \Pi_H \). Thus, there is a corner case where quality competition does not occur that is characterized by \( q_L = \arg \max_{q_H} \Pi_H \). Solving for \( k \) yields

\[ \tilde{k} = \frac{N_1}{N_1 + \sqrt{N_1^2 - N_2^2}}. \]  
(23)

Consequently the \( H \) developer’s maximum quality level is:

\[ q_H = \begin{cases} \arg \max_{q_H} \Pi_H = \frac{(1-s)N_1}{2k} & \text{for } k \leq \tilde{k} \\ q_L & \text{for } k > \tilde{k} \end{cases} \]  
(24)
This means that two types of equilibria can arise depending on $k$. First, a pure strategy equilibrium occurs if $k \leq \tilde{k}$. In this case $H$’s maximization problem results to

$$q_H = \arg \max_{q_H} \Pi_H = \frac{(1 - s) N_1}{2 k},$$  \hspace{1cm} (25)

and $L$’s maximization problem yields

$$q_L = \arg \max_{q_L} \Pi_L = \frac{(1 - s) N_2}{2}.$$  \hspace{1cm} (26)

This is an unique equilibrium since choosing $q_L > q_H$ yields lower profits for $L$ due to $q_H \geq \overline{q_L}$.

Second, a mixed strategy equilibrium can arise for $k > \tilde{k}$. Suppose the $H$ developer chooses $q_H \in (q_H, \overline{q_H}]$ with any positive probability $p$. Then there is a positive probability for a tie if choosing the quality level $q_H$ is part of $L$’s quality strategy. Note that $L$’s profit in case of a tie is $(1 - s) q_L N_2 - C_L$. A slightly higher quality yields a marginal increase of costs for $L$, but a significant increase of expected revenue, since its probability to attain $N_1$ users increases by $p$. Consequently choosing the quality level $q_H$ cannot be part of the equilibrium strategy of $L$ if $q_H$ is chosen by $H$ with probability $p$. Consider further that if $q_H$ is not played by $L$, a marginal lower quality level yields the same probability for $H$ to attain $N_1$ users. Thus, choosing $q_H$ minus an increment is strictly better than choosing $q_H$ with probability $p$ as long as the decrease in costs for quality is higher as the decrease in expected revenue. Choosing the lower bound $q_H$ with probability one is also not an equilibrium, since $L$ would outbid $H$ by choosing $q_H$ plus an increment. Consequently, an equilibrium quality strategy of $H$ consists of a mass point at its lower bound, and an atomless distribution of $q_H$ between $q_H$ and $\overline{q_H}$.

The equilibrium strategy of $L$ looks similar. We already mentioned that $L$ will never choose a quality that is played by $H$ with probability $p$. Thus, it will never play $q_H$ as long a $q_H$ is a mass point in $H$’s equilibrium strategy. It will further choose no quality level between
and $q_L$, because the probability to attain $N_1$ users remains zero in this range; but the profit from attaining $N_2$ users is maximized at $q_L$. Further, $L$ will never choose a $q_L \in (q_H, q_L]$ with a positive probability $p$. To see this, suppose $L$ chooses a quality $q_L$ with probability $p$. Then $H$ will never choose a $q_H$ which is equal to $q_L$ minus an increment, because the probability to attain $N_1$ users is much higher by choosing $q_H = q_L$. But, if $H$ never chooses $q_L$ minus an increment then choosing $q_L$ minus an increment instead of $q_L$ yields the same probability for $L$ to attain $N_1$ users. Consequently, an equilibrium strategy of $L$ can only have a mass point at $q_L$. This mass point is always part of L’s equilibrium strategy. If $L$ never chooses $q_L$ then $H$ always loses the quality based competition by choosing $q_H$. Note that in such a situation $H$ gets $\tilde{\Pi}_H$ if it chooses $\overline{q}_L$, and $(1 - s) q_H N_2 - k q_H^2$ if it chooses $q_H$. Note further that for $k = 1$ both profits are equal, because the equality of both profits determines the maximum quality level that $L$ chooses. For every lower $k$ the revenue side of both profits remains equal, but the costs of choosing $\overline{q}_L$ decrease by a factor $\overline{q}_L^2$, and the costs of choosing $q_H$ decrease by a factor $q_H^2$. That is, for every $k < 1$ choosing $\overline{q}_L$ yields a higher profit. Thus, $H$ will never choose a quality level which is lower than the minimal quality level of $L$. Consequently, any strategy where $L$ does not choose $q_L$ cannot constitute an equilibrium. Finally, choosing $q_L$ with probability one cannot constitute an equilibrium either, because the best response of $H$ is to choose $q_H$ with probability one, which, however, cannot be an equilibrium strategy, as already mentioned.

Thus, equilibrium strategies of both developers consists of a mass point at the respective lower bound, and an atomless distribution between $q_H$ and $\overline{q}_L$. Consequently, for every quality choice of $L$, the condition $\Pi_L \overset{1}{=} \tilde{\Pi}_L$ holds, and for every choice of $H$, the condition $\Pi_H \overset{1}{=} \tilde{\Pi}_H$ holds. Solving for CDFs yields:

$$F_H = \frac{(q_H - q_L)^2}{(1 - s)(N_1 - N_2) q_H},$$

(27)
Finally, with the CDFs at hand, we can derive $q_H$. Let the corresponding probability density functions (PDF) of $H$’s and $L$’s quality choice be $f_H$ and $f_L$, respectively. Values of $q$ where one of the PDFs is negative cannot be reasonable. As both PDFs are increasing in $q$, we can solve for the minimum feasible $q$ by solving for the maximum root of both PDFs. Solving $f_L = 0$ and $f_H = 0$ we obtain that the maximum root is

$$q_H = \max \left\{ \sqrt{\frac{\tilde{\Pi}_H}{k}}, \sqrt{\tilde{\Pi}_L} \right\} = \sqrt{\frac{\tilde{\Pi}_H}{k}}. \quad (29)$$

Note that this minimum quality is chosen by the $H$ developer with a probability of $F_H(q_H)$. Likewise, the probability that the $L$ developer chooses its minimum quality $q_L$, and does not compete for the top position in the ranking, is $F_L(q_H)$.

### B  Proof of Lemma 2

We start by calculating the reasonable payment range for each developer. The outside option of $L$ is to bid $r_L = 0$. Consequently, the minimum reasonable payment of $L$ is $\underline{r} = 0$. By choosing this payment, it obtains an outside option profit of

$$\tilde{\Pi}_L = (1 - s) q_L N_2 - C_L \quad (30)$$

from serving $N_2$ users. Note that $\underline{r} = 0$ is also the minimum reasonable payment of the $H$ developer, since it is sufficient to match $L$’s bid to win the competition. Assume that there is a payment, $\overline{r}$, for which the wining probability is one, so that the $L$ developer is indifferent between choosing $\underline{r}$ and $\overline{r}$. This maximum reasonable payment of the $L$ developer can be
obtained by solving $\tilde{\Pi}_L \doteq \Pi$, which results to

$$r = (1 - s) q_L (N_1 - N_2).$$  \hfill (31)

Again, recall that this is also the maximum reasonable payment of $H$ since it can ensure to get the top position simply by bidding $r_H = \bar{r}$. We denote this as the outside option of $H$, which yields a profit of

$$\tilde{\Pi}_H = (1 - s) q_H N_1 - C_H - \bar{r}. \hfill (32)$$

Then there is a positive probability for a tie if choosing the quality level $q_H$ is part of $L$’s quality strategy.

Based on this we can proof the existence and uniqueness of the mixed strategy equilibrium. Suppose $H$ chooses a payment $r_H \in (0, \bar{r}]$ with a positive probability $p$. Consequently there exists a positive probability for a tie if bidding the amount $r_H$ is part of $L$’s bidding strategy. Note, that $L$ attains only $N_2$ users in case of a tie and can significantly improve its probability to attain $N_1$ users by bidding $r_L$ plus an increment. Consequently, it is never reasonable to bid $r_H$. If $L$ never bids $r_H$, then bidding $r_H$ minus an increment yields the same probability to win, but at a lower cost. Thus, bidding $r_H \in (0, \bar{r}]$ with positive probability cannot be an equilibrium strategy. Obviously bidding zero with probability one is also not an equilibrium strategy, because $L$ would outbid $H$ by bidding slightly more. The equilibrium strategy of $H$ must consist of an atomless distribution in the range of $r_H \in (0, 1]$ and can have a mass point at zero.

The equilibrium strategy of $L$ is also an atomless distribution in the range of $r_L \in (0, \bar{r}]$ with a mass point at zero. To see this, consider that $L$ bids $r_L$ in this range with a positive probability $p$. In this case it is never reasonable for $H$ to bid $r_L$ minus an increment, because this yields marginal cost savings, but a significant decrease in the probability to attain $N_1$ users. But, if $H$ never bids $r_L$ minus an increment, then bidding $r_L$ minus an increment
yields the same probability for $L$ to attain $N_1$ users, but at a lower cost. Thus, a strategy with a mass point in the range $(0, \bar{r}]$ can never be an equilibrium strategy. Bidding $r_L = 0$ with probability one is obviously also not an equilibrium strategy, because the best response of $H$ would be to bid zero with probability one, which as already mentioned, cannot be an equilibrium strategy.

The mixed strategy equilibrium from above can only exist if $\tilde{\Pi}_L \equiv \Pi_L$ and $\tilde{\Pi}_H \equiv \Pi_H$, which results in the following CDFs:

$$G_H = \frac{r_H}{(1-s)q_L(N_1-N_2)},$$  \hfill (33)

$$G_L = \frac{1}{(1-s)q_H(N_1-N_2)}(r_L - (1-s)((N_1-N_2)q_H - (N_1-N_2)q_L)).$$  \hfill (34)

### C Proof of Lemma 3

Note that the expected profit of each developer is equal to its outside profit, regardless of what is the particular realization of its bidding strategy. Consequently, the developers’ maximization problems result to

$$q_L = \arg \max q_L \tilde{\Pi}_L = \frac{(1-s)N_2}{2},$$  \hfill (35)

$$q_H = \arg \max q_H \tilde{\Pi}_H = \frac{(1-s)N_1}{2k}.$$  \hfill (36)

### D Proof of Proposition 4

The first part of the proposition follows immediately from Figure 6, which provides a complete characterization of the comparison of app qualities between a quality-based ranking and a sponsored ranking. For the second part of the proposition, to see that $\Delta^s = q_H^s - q_L^s > \Delta^q = q_H^q - q_L^q$, recall that $s^q = \frac{1}{2} > s^s$ over the whole range of parameters. Thus, $(1-s^s)-(1-s^q) > 0$. For $k \leq \bar{k}$ it follows that $\Delta^s - \Delta^q = ((1-s^s)-(1-s^q))(N_1/2k-N_2/2) > 0$. 

53
Let $\gamma = 1$ for now. For $k > \tilde{k}$ as $k \to 1$, due to the competition effect, it follows that $\Delta^q$ decreases more than $\Delta^s$. Eventually, at $k = 1$, we already know that $\Delta^q = 0$, whereas $\Delta^s > 0$. Thus, we have shown that $\Delta^s > \Delta^q$ when $\gamma = 1$. Now consider what would change when $\gamma < 1$. Clearly, this has no impact on $\Delta^s$. However, when $\gamma$ decreases, $\Delta^q$ will decrease as well (see Proposition 2). Thus, $\Delta^s > \Delta^q$ must hold for all values of $\gamma$. 

54