**All of the same breed? A networking perspective of private-collective innovation**

## ABSTRACT

Much research on open source software development has highlighted the best-of-both-worlds benefits generated by private and collective contributions from a broad base of developers. However, these studies have tended to overlook the heterogeneous nature of various developer groups including firm-sponsored developers. To unveil the behavioural differences between developer groups, we expand upon the private-collective innovation model using a networking approach, linking network closure and positional embeddedness with technical contribution. We tested our predictions using a lagged analysis based on communication and networking behaviours on the Linux kernel mailing-list in the production of the Linux operation system over a three-year period. Our findings support our predictions, showing that network closure has an adverse impact on technical contribution. Additionally, the relationship between positional embeddedness and technical contribution follows an inverted U-shape. In addition, high positional embeddedness counteracts the negative influence of extensive network closure on technical contribution. These effects are partly moderated by respective developer groups. Our model and results offer important theoretical and practical implications for community management within the framework of private-collective innovation.

**Keywords:**

Network theory, Open Source Software, Private-Collective Innovation, Propensity score matching

## 1. Introduction

Open source software (OSS) development has undergone a significant evolution, transitioning from being primarily driven by volunteer developers to receiving substantial sponsorship and support from hardware and software companies (Fang & Neufeld, 2009; Von Krogh et al., 2012; Qu et al., 2011). A prime illustration of this shift is the Linux project, which has changed software development and usage across diverse communities and industries. This collaborative software co-production exemplifies *private-collective innovation*, wherein a diverse range of actors, each with their private interests[1], collaborates to yield a collective output (Gächter et al., 2010; Ramaswamy & Ozcan, 2018; Von Hippel & Von Krogh, 2003). What makes private-collective innovation unique is that developers freely reveal their "proprietary information" embedded within source codes (Henkel, 2006; Von Hippel & Von Krogh, 2006). Consequently, private-collective innovation refers to situations where private investments contribute to the production of a public good (Erickson, 2018). This public good can be combined with other private goods and marketed as a commercially viable product. Hence, while the collective output's intellectual property remains public, sponsoring firms can still derive their unique *private* benefits.

One prevailing sponsorship model entails seconding internal software developers from firms to engage in OSS communities, enabling these firms to align OSS development with their present and forthcoming proprietary products (Schaarschmidt et al., 2015). In this regard, technical contributions such as source code commits from firm-sponsored developers serve as an effective proxy for the unique private benefits that arise from developing proprietary software using open source approaches. Nonetheless, tensions may arise between public interests and

[1] We note that both sponsored developers and volunteers have private interests, thus, private interest does not equal commercial interests.

firm-based interests, particularly when software and hardware companies possess differing visions for the same OSS project. However, prior studies suggest that developers tend to self-organize "by building a bridge between divergent worlds that allows collaborators to preserve their competing interests" (O'Mahony & Bechky, 2008, p. 426). Self-organizing underscores the significance of strategic interaction and networking within OSS communities (Kuk, 2006). Developers from various affiliations, including volunteer, educational, kernel[2] developers, and firm-sponsored developers, tend to self-select projects (Santos et al. 2013) that aligns with their values and/or interests (Maruping et al., 2019). How this self-selection translates into varied networking behaviour within the context of private-collective innovation remains largely unexplored. Given this self-selection process involving a wide array of developers, it is unlikely that all developers initiate on an equal footing, with the majority volunteering and only a few being financially supported by their sponsors. Thus, rather than treating developers as a uniform group with cohesive interests (e.g., Temizkan & Kumar, 2015; Toral et al., 2010), it is plausible that different developer groups may act according to their distinct interests, thereby leading to varying networking behaviour. Despite considerable research on networking among OSS developers as a uniform group (e.g., Chou & He, 2011; Dahlander & O'Mahony, 2011), the pertinent literature remains relatively silent about the dissimilarities in networking behaviour across diverse developer groups. Consequently, our primary theoretical and empirical objective is to unravel the connection between networking behaviour and technical contribution (measured through source code commits) across distinct developer groups in the context of private-collective innovation.

[2] Kernel developers refers to developers that use a "kernel.org" email-address in the Linux kernel mailing list. Typically, they are assigned to the Linux foundation.

Addressing this research void is paramount for several theoretical and practical reasons. Firstly, from a theoretical standpoint, dominant imageries such as core and periphery (Lee & Cole, 2003), the onion structure (Masmoudi et al., 2009), and a "man from inside" (Dahlander & Wallin, 2006) have generally downplayed the relational dimensions of networking behaviour, particularly between kernel and firm-sponsored developers. Additionally, in the context of private-collective innovation, there is an assumption that participation yields unique benefits, such as learning from innovation communities (Henkel, 2006). Whilst existing evidence suggests an array of individual and social benefits, the ways sponsored developers derive commercial advantages through networking remain underexplored.[3] Firm-sponsored contributions can potentially establish path-dependent knowledge, granting sponsoring firms an expedited path to vertical markets by combining the collective output with the firm-based co-specialised assets and complementarities (Alexy & Reitzig, 2013).

Second, from a practical standpoint, delving into networking behaviour allows us to examine how sponsoring firms harness private-collective innovation to gain a distinctive competitive advantage. With such a stake for sponsoring firms, this also directs our attention to examine the autonomy assumption that sponsored developers remain independent from their sponsoring firms. It is likely that sponsored developers may engage in distinct networking behaviours compared to their non-sponsored counterparts. The above theoretical reasoning and practical relevance motivates our research by asking, "how do firm-sponsored developers influence technology development through networking, and how does this association differ from other non-sponsored developer groups?"

[3] We note that economic motivations for firms to participate in OSS development has been widely studied (e.g., Bonaccorsi et al., 2006; Henkel, 2006). Our focus is on networking behavior of firm-sponsored developers in contrast to other developer groups. We thank one reviewer for the suggestion to clarify our goal in this regard.

To answer this question, we delve into the networking behaviours exhibited by various developer groups linked to their technical contribution over a three-year period. This timeframe allows us to assess our model's validity by scrutinizing the communication and interactions of over 11,000 contributors on the Linux mailing lists. We then correlate these interactions with their subsequent technical contributions in terms of committed source code in the subsequent year. Consequently, our research carries significant implications for both theoretical understanding and the management of private-collective communities. Firstly, our approach offers a solid theoretical foundation for understanding how actors with private interests collaborate to achieve a shared objective of producing a commercial-grade and freely available OSS product while simultaneously pursuing activities that matter to individual contributors. It also allows us to extend the private-collective innovation by focusing on resource needs that bring developers of divergent interests to form network ties through a free revealing process (Henkel, 2006). By doing so, we provide a networking perspective of private-collective innovation by gauging how pockets of networking behaviour transcend into network structures, which subsequently influence technical contribution. Secondly, viewed from a managerial perspective, our research stands to assist developers actively engaged in private-collective innovation settings. It aids them in identifying the networking behaviour mostly likely to realize private benefits achieved through source code commits.

In the next sections, we first introduce the conceptual background. After that, we develop a set of research hypotheses linking networking across different contributing groups with technical contribution. We then continue with the empirical part that draws upon multi-sourced data to test our model. Finally, we discuss our results with reference to the extant literature of OSS development and social network theory.

## 2. Theoretical background

### *2.1. Private-collective innovation*

From an innovation economics view, three types of incentive structures define the relationship between private and collective actions, and the conditions when private and collective actions can co-exist in an open innovation setting (Gächter et al., 2010). Firstly, in a strictly private innovation model, incentives are designed in a way that private action remains independent of collective action by avoiding knowledge spillovers and free revealing through the use of intellectual property right protection (Alexy et al., 2013; Alexy et al., 2018). In contrast, the collective model hinges on virtuous incentive, driven by the production of public goods. These public goods possess the attributes of non-excludability and non-rivalry, thus eradicating the incentive for private action to produce them as a club good (Benkler, 2013; Walter et al., 2007). Lastly, the private-collective model offers a middle-ground solution to accommodate private and collective interests through a hybrid incentive structure. This hybrid approach, characteristic of the private-collective innovation model (Alexy & Reitzig, 2013; von Hippel & von Krogh, 2006), allows private entities, including software firms, to openly share their innovation at their own expense, in exchange for unique private benefits that can only be derived through collective action. This amalgamation of market and community logics underscores the foundation of the private-collective innovation model. In this model, software firms willingly reveal their innovation to obtain private benefits that arise exclusively through collaborative efforts. These private benefits encompass various aspects, including process-related advantages such as learning from the community (Von Hippel & Von Krogh, 2003). In addition, for software firms, the availability of a publicly accessible commercial-grade product holds strategic significance, delivering substantial value within vertical markets.

*2.2. A networking perspective of technical contribution*

Ensuring that competitiveness and security of OSS applications holds, while consistently aligning with user needs and not burdening sponsoring firms, hinges on technical contribution (Daniel et al., 2018; Maruping et al., 2019). As such, studying technical contribution becomes imperative. Technical contributions can take various forms, including offering technical suggestions through email lists, altering the number of files, or making source commits to a shared repository (Daniel et al., 2018). In this study, we designate source code commits as a proxy for technical contributions for two reasons: 1) It has been used as such in previous research (e.g., Maruping et al., 2019), and 2) source code commits enable firms to reduce costs by facilitating integration with proprietary code.

While it is important for an OSS project to rely on competent developers with the necessary skill sets, OSS development also entails a distinct form of networking behaviour (Kuk, 2006). This networking behaviour is recognized to influence the flow of knowledge, stimulate creativity, and ultimately drive innovation (Resch & Kock, 2020, Singh & Tan, 2010; Von Krogh et al., 2012). Many scholars have invoked social capital theory to elucidate the emergence of technical contribution, such as code commits (e.g., Grewal et al., 2006; Maruping et al., 2019; Tang et al., 2020). This structural approach to understand technical contribution has reinforced the importance of developers occupying central network positions (e.g., Dahlander & O'Mahony, 2011). In particular, associating central network positions with exclusive access to knowledge and information creates an information asymmetry that benefits developers near to the core over those situated at the periphery (Burt, 2017).

However, debates persist regarding the effects and methods of measuring social capital (Burt, 2017). Specifically, despite the extant use of social network theory in OSS development,

three major research gaps have emerged. Firstly, the majority of OSS studies have leaned heavily towards structural network measures, such as indegree and outdegree centrality, which respectively gauge the number of *direct* connections the focal actor receives (indegree) and the number of *direct* peers that are communication targets of the focal actor (outdegree) (Dahlander & Wallin, 2006; Maruping et al., 2019). As shown in Appendix A, most articles draw on the structural centrality measure. While these measures are valuable, they tend to overly emphasize the quantity of connections while neglecting their quality, specifically the interconnections among an individual's peers. Secondly, with few exceptions (e.g., Dahlander & Wallin, 2006, Grewal et al., 2016; Tan et al., 2015), research has largely ignored the simultaneous consideration of various networking behaviours. It is crucial here to differentiate between the impact of being connected to well-connected actors (i.e. positional embeddedness) and being linked to actors who share connections with the same set of the actors as the focal actor (i.e. network closure) (see 2.3). Lastly, the network perspective within OSS development has often treated community actors as a largely uniform group, thus downplaying developing software that is straddled between community structures and organizational hierarchies (Di Gangi et al., 2022; Schaarschmidt, 2022; Teigland et al., 2014). Specifically, in the context of private-collective innovation, contributors do not all participate for the same private reasons (Shah, 2006). Firms, for instance, tend to engage in projects that offer strategic benefits (Shaikh & Levina, 2019; Spaeth et al., 2015), while volunteers are driven by a desire to learn from proficient developers and enhance their software development skills. Despite these inherent distinctions, technical contribution remains pivotal in highlighting how both volunteer and sponsored developers attain their distinct benefits through diverse networking behaviour.

*2.3. Network closure and positional embeddedness*

We draw upon the literature on social network theory to explore the links between networking and technical contributions across different developer groups (e.g., Chou & He, 2011; Dahlander & O'Mahoney, 2011). Our approach diverges from conventional networking measures such as in- and out-degree (see Dahlander and Wallin, 2006), and instead focuses on by two distinct networking dimensions that account for the quality of connections. Specifically, we investigate networking measures in the forms of network closure (evaluated by the density of interconnections within a developer's network is interconnected, measured through aggregate constraint, Gargiuli et al., 2009); and positional embeddedness (assessed by a developer's connections with other well-connected developers, using eigenvector centrality, Grewal et al., 2006). These networking measures mirror different aspects of social capital that we will detail next.

High network closure refers to a situation where an individual is closely linked to a group of peers, who themselves maintain strong interconnections (Tortoriello et al., 2015). Prior research has highlighted this as a two-edged concept (Gargiuli et al., 2009). On one hand, network closure aims to foster trust among network members within the network (Coleman, 1990). Conversely, it can lead to knowledge redundancy, with network members sharing and accessing similar knowledge that contributes little to innovation and the collective outcomes (Aral, 2016). Therefore, while high network closure promotes, it also exposes participants to the drawback of redundant knowledge.

In contrast, networking that enhances positional embeddedness can be selectively directed by connecting with influential peers or projects (Kuk, 2006). Positional embeddedness has been defined as "*the extent to which an entity is connected with other structurally embedded entities*" (Grewal et al., 2006, p. 1045). High positional embeddedness can potentially grant a

developer enhanced influence and a more exclusive position within a smaller network of influential peers. Grewal et al. (2006) have demonstrated that OSS projects marked by high network embeddedness tend to achieve greater technological advancement and commercial success. Therefore, we anticipate a similar relationship between positional embeddedness and technical contribution in the context of private-collective innovation.

## 3. Hypotheses

OSS technical contribution provides developers with the means to realise their unique private benefits through co-producing high-quality commercial products. Irrespective of their affiliations or incentives, technical discussions entail the sharing and exchanges of new information and knowledge, all while deliberating on developmental priorities. As a result, networking – manifesting through exchanges of ideas and deliberation – acts as a mechanism to broaden the availability and dissemination of new technical knowledge. The following hypotheses are formulated based on the premise that networking is central to technical contribution driven by shared resource needs inherent in the context of private-collective innovation (Resch & Kock, 2020).

### *3.1. Network closure and technical contribution*

As previously discussed, high network closure offers both advantages and disadvantages. While it fosters a sense of trust that binds network members (Allcott et al., 2007), it can also restrict the potential for leveraging structural holes (Borgatti et al., 2018; Burt, 2004). This often results in sharing the same redundant knowledge (Kuk, 2006; Singh et al., 2011). Although high network closure is essential to facilitate deliberation and garnering support from volunteers who may have reservations about the sponsorship (Gargiulo et al., 2009; Spaeth et al., 2015), its drawbacks tend to overshadow its benefits in a private-collective context. For example, in such a

context, deliberation may encompass decisions about prioritizing priority new plug-ins and upgrades involving proprietary hardware and software from sponsoring firms. These decisions may take significantly longer for firm-sponsored developers to disseminate additional and fresh information, and to engage with other developer groups to deliberate on these priorities (Aral, 2016). Consequently, this could lead to a deceleration in the pace of technical contributions. Thus, under equal circumstances, networking that results in high network closure within a private-collective context is likely to dampen technical contribution. This has led to the following hypothesis.

*Hypothesis 1 (H1): Network closure is negatively associated with technical contribution in the form of source code commits.*

In a private-collective context, the validity of H1 may not hold across all developer groups. This suggests that the extent of the negative effect stemming from high network closure on technical contributions could vary among different developer groups. The literature on open-source communities suggests at least five distinctive developer groups (e.g., Singh & Tan, 2010; Ye & Kishida, 2003). Each brings a unique set of knowledge and resources to the community. They are: 1) firm-sponsored developers, which include developers seconded from sponsoring firms; 2) kernel developers who are responsible for administrative and coordination roles; 3) education developers represent university employees and develop tailored software for higher education; 4) volunteers who are active developers without any firm affiliation; and 5) users who passively engage with the OSS development and contribute their need knowledge in the OSS development. In formulating our next hypotheses, we use firm-sponsored developers as the reference group to contrast their networking and technical contribution with the other developer groups.

Networking involves developers opting to join discussions based on their personal interests, and those of the organizations that they are affiliated with. Despite its dampening influence on technical contribution (as predicted in H1), high network closure offers the advantage of making technical information more easily to comprehend (Ter Wal, Alexy, Block, and Sander, 2016). For firm-sponsored developers, their primary concern may revolve around specific sections of the software aligned with their sponsors' interests (Schaarschmidt, 2023). Networking facilitates the sharing and translation of proprietary knowledge, ensuring that any changes in code production align with the present and future interests of the sponsoring firms. Consequently, networking among firm-sponsored developers tends to be more focused and purposeful, aimed at garnering broader support from other developer groups. Additionally, firm-sponsored developers are likely to dedicate more time to OSS projects than volunteers (Riehle et al., 2014) and engage in deeper conversations (Aral, 2016). This heightened level of engagement increases the complexity of exchanges and deliberations, potentially making networking more cumbersome for firm-sponsored developers and exacerbating the adverse impact of high network closure on technical contribution. In contrast to other developer groups, the negative effect of network closure on technical contribution is anticipated to be more pronounced for firm-sponsored than other developer groups. This leads us to formulate the following hypothesis:

*Hypotheses 2 (H2): The negative relationship between network closure and technical contribution is moderated by the types of developer groups, such that the negative relationship is stronger for firm-sponsored developers compared to other developer groups.*

### *3.2. Positional embeddedness and technical contribution*

The second networking behaviour is captured through positional embeddedness, which indicates the degree to which a developer is connected with other well-connected counterparts

(Packard et al., 2016; Zukin & DiMaggio, 1990). A high level of positional embeddedness signifies a closer connection to the central group of developers and provides increased opportunities for making valuable technical contributions. Prior studies have shown the positive effect of positional embeddedness on innovation outcomes (e.g., Grewal et al., 2006) by facilitating access to a larger resource pool (Grewal et al., 2006; Owen-Smith & Powell, 2004; Provan & Lemaire, 2015). Consequently, well-connected developers are more sought after than those with fewer connections, as they can facilitate access to other valuable resources, further expanding their networks of well-connected developers. Hence, high positional embeddedness can enhance the realisation of unique private benefits in the form of source code commits. However, there is a point when positional embeddedness reaches an optimal level, having numerous connections to too many well-connected individuals can lead to heightened maintenance demands, potentially outweighing the information benefits (Laursen & Salter, 2006). Consequently, once this optimal threshold is surpassed, the initial positive relationship between positional embeddedness and technical contribution takes on a negative trajectory.

*Hypotheses 3 (H3): The relation between positional embeddedness and technical contribution follows an inverted U-shape, such that higher positional embeddedness is associated with more technical contribution, but only up to a certain point after which more positional embeddedness has negative effects.*

However, the influence of positional embeddedness on realising distinct benefits through technical contribution can diverge among various developer groups. In the case of firm-sponsored developers, who dedicate their full time to OSS projects, a unique dynamic is at play. They are inclined to form ties with their peers due to their shared business-oriented resource requirements and technical challenges. For example, two firm-sponsored developers may be both

invested in addressing technical compatibility issues related to integrating OSS with proprietary hardware and software in specific vertical markets (Linåker et al., 2019). Consequently, firm-sponsored developers tend to foster connections among themselves. Additionally, given their similar needs, their networks are more likely to be interconnected than a network of volunteers, who often engage sporadically and typically hold peripheral positions in the network with limited access to influential others.

Thus, when non-sponsored developers establish ties with well-connected counterparts, it offers them greater pathways to contribute technically. In essence, once a non-sponsored developer (such as a kernel developer, education developer, volunteer, user) successfully forges ties with well-connected others, the relative advantage becomes more pronounced compared to firm-sponsored developers. These structural distinctions explain why non-sponsored developers are inclined to reap more benefits from positional embeddedness than their firm-sponsored counterparts.

*Hypotheses 4 (H4): The relationship between positional embeddedness and technical contribution is different for different developer groups, such that firm-sponsored developers' technical contribution (compared to user, volunteer, kernel, and education) is less affected by high positional embeddedness.*

### *3.3. Network closure and positional embeddedness*

Lastly, we explore the combined effect of high network closure and positional embeddedness. The challenge with high network closure lies in the potential inundation of developers with information they are already familiar with. This could impede the network's ability to create novel solutions, especially with homogenous networks. However, in private-collective innovation, high network closure can encompass members who also hold high

positional embeddedness. Essentially, within a heterogeneous network, the collaboration of well-connected developers working closely together can leverage their positional embeddedness to counteract the negative effect of high network closure on technical contributions.

In OSS development, the well-connected developers (i.e. those with high positional embeddedness) often undertake coordination tasks than focusing on writing codes. For instance, in a study of GNOME, Dahlander & O'Mahony (2011) reveal that core developers who were elected to the board of directors tended to engage more in administrative and coordination duties than direct technical contribution. Essentially, high network closure centred around these core developers, who are inherently well-connected can provide valuable 'borrowed social capital' to external parties (Burt, 2004). This suggests that a high network closure encompassing well-connected developers serves as more than just a source of positional power, it also facilitates bridging among different developer groups, fostering consensus in support of technical contribution (O'Mahony & Bechky, 2008). Thus, with the presence of well-connected developers, high network closure can yield positive effects. We therefore propose the following hypothesis:

*Hypothesis 5 (H5): The combined effect of high network closure and positional embeddedness on technical contribution is positive.*

Figure 1 displays the conceptual model underlying our set of hypotheses.

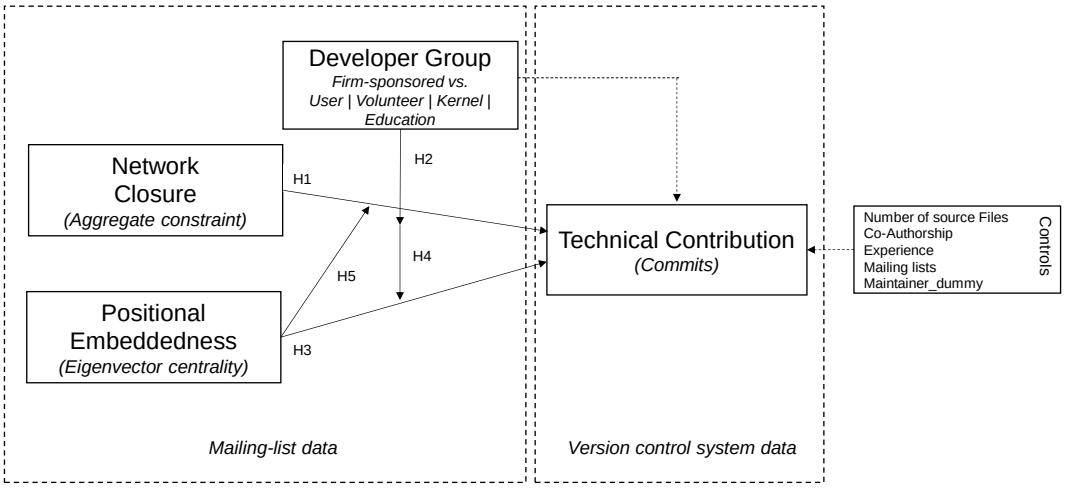


**Figure 1.** Conceptual model

## 4. Method

### *4.1. Sample and data collection*

To study the relationship between network closure, positional embeddedness, developer groups, and technical contribution within the context of private-collective innovation, we conducted a historical social network study focusing on the Linux kernel project. The Linux kernel project serves as a prime example of a large private-collective endeavour (Germonprez et al., 2017), characterised by community initiation and firm-driven collaboration among multiple vendors (Schaarschmidt, 2023). Historical case studies prove valuable when specific events trigger notable actions (Runfola et al., 2017). In this regard, we selected a period from 2006 to 2008 for analysing communication networks within mailing lists. This time frame aligns with the 2006 release of a major Linux kernel version (Linux kernel 2.6), marking the rise of increased

company involvement (cf. Homscheid et al., 2015). Consequently, the project experienced a transition phase during this period. This approach is particularly fitting as our primary goal is theory testing. For our network data, we chose a three-year time span for two reasons. Firstly, a three-year duration is reasonable for initiating and fostering relationships within a communication network. Shorter time frames might underestimate relationship development, while longer ones could add complexity. Our approach thus aligns with recent network studies employing archival data (e.g., Foss et al., 2020).

Linux, an open source operating system built on the UNIX[4] foundation, has evolved into multiple derivative forms (Lee & Cole, 2003; Schaarschmidt, 2012). In stark contrast, the Linux kernel represents the system's technical core, housing essential processes and functions (such as hardware control and graphic processing). As a result, the Linux kernel serves as the foundation for a wide range of Linux distributions, including Debian, Fedora, and Ubuntu, as well as for embedded systems and mobile operating systems like Android. The Linux kernel project stands out for its robust activity and remarkable success. Despite Linux's significance for various firms and their business strategies, it is surprising that between stable release 4.8 and 4.13, approximately 12.3% of all source code changes were contributed by individuals without any known company affiliation in Linux development (Corbet et al., 2017). It is worth noting that the majority of developers working on the kernel receive compensation for their contributions. Notabnly, among the prominent contributors between versions 4.8 and 4.13 were Intel (13.1%), RedHat (7.2%), and IBM (4.1%). However, firms focused on the mobile operating systems market made relatively modest contributions to the Linux kernel with, for instance, 3.0% (Google) and 3.2% (Samsung) (Corbet et al., 2017).

[4] It is important to note that although Linux's architecture is oriented toward UNIX, Linux does not contain any UNIX code.

For this study, we combined multiple sources of information into one database. We collected historical email data from the Linux kernel mailing list (see Dahlander & Wallin, 2006; Kuk, 2006; Schaarschmidt, 2012), along with source file data and details about lines of accepted software code (both inserted and deleted) from the Linux's Concurrent Versions System (CVS) named Git. These data components – mailing list records, source files, and lines of code – offer objective measures that are less susceptible to the influence of subjective reconstruction (Aral, Brynjolfsson & Van Alstyne, 2012).[5] Moreover, this comprehensive approach helps us avoid the selection bias commonly encountered in survey-based network studies, as we are using a complete network dataset (Dahlander & Wallin, 2006).

To ensure comprehensive analysis, we collected mailing list data spanning from 2000 to 2008. This broader timeframe allowed us to include control variables such as experience, which could influence our network measures calculated for our main time frame of interest, 2006 to 2008. The data was acquired from marc.info, a platform that systematically mirrors the relevant Linux mailing lists. Throughout the entire period between 2000 and 2008, over 100,000 individuals contributed at least one message to these mailing lists. However, our specific data subset focused on the 11,899 individuals who had posted at least one message to the Linux kernel mailing list between 2006 and 2008. For these individuals, we further gathered data from the source file of Linux version 2.6.29.4, which captured source code authorship until the beginning of 2009. The choice of the initial version of 2009 was deliberate, as not all commits directly result in a new version. Finally, our data collection involved gathering information on lines of code accepted (covering both insertions and deletions) from Git for all 11,899

[5] In a laborious process, data was cleaned to exclude "Hello"-messages (and similar), Emails from bots such as *mailer-deamon*, duplicate messages received from external mailing lists, and various forms of spam (e.g., Viagra advertisement) (see Online Appendix B)

individuals during the year of 2009. This data was linked to their respective email-addresses and used as the dependent variable for our analysis.

### *4.2. Measures*

#### 4.2.1. Dependent variable

Our unit of analysis was at the individual level. We measured our dependent variable – technical contribution – using the lines of code (which encompass both lines inserted and lines deleted), that individual developers contributed to the CVS code repository, which is Git in the case of Linux. Each set of lines of code submitted was referred to as a "commit". Using lines of code offer a more detailed measure of technical contribution in contrast to commits, which can vary significantly in size (Mauping et al., 2019). Not only new lines of source code are technical contribution; deleted lines of source code also hold value, such as in the maintenance or removing outdated features (Lotufo et al., 2010). We aggregated line inserts and line deletes for each source code commit, resulting in a number representation of the changes made by a particular source code author in a specific commit.

Our dataset consisted solely of lines of codes from commits that had been incorporated into subsequent releases of the Linux kernel after undergoing review. Commits that were uploaded to Git but which were not accepted as part of the official Linux kernel version was excluded. This approach had two benefits: 1) it allowed us to capture the developer's private benefit derived from accepted code, which holds higher value compared to unaccepted code, and 2) it indirectly served as a control for code quality, as excessively lengthy or unnecessary code would be less likely to be accepted.

For our analysis of technical contributions, we opted to focus on the year 2009. This choice was informed by our aim to regress technical contribution in 2009 against network

measures spanning 2006 to 2008. Our decision considers the possibility that the knowledge acquired from networking may not immediately translate into technical contributions, requiring some time for discussions on the mailing lists to transform into new code commits. As a result, we gathered data on the number of insertions and deletions made by 11,899 individuals in our dataset during the year 2009. In our final sample, 1,111 (9.34% of the total) had submitted at least one commit in the year 2009.

4.2.2. Independent variables

All independent variables we considered are related to communication behaviour within the Linux kernel mailing list. Communications on this mailing list is organised into threads that are grouped by months. Our software program was designed to evaluate each message within a thread to determine if it was directed to a specific individual indicating by phrases like “Dear Ralph”. This analysis was feasible because most messages on the Linux kernel mailing list adhere to a quasi-standardized format (RFC1855). When a message contained a specific addressee, we established a directed link between the sender and the receiver of the message. in instances where a message lacked a specific addressee, the program inferred that it was relevant to the sender of the previous message in the same thread. In such cases, we established a link between the sender and the sender of the preceding message. By implementing this process, we were able to construct a communication matrix spanning three years and encompassing all 11,899 developers. To address the limitations of mailing list data, such as instances where individuals used multiple email addresses (Howison et al., 2011), we employed a *mapping table*. This mapping table allowed us to recognize each virtual identity of a developer in the mailing list data (as well as the source file data) and consolidate them into a single real-life identity. The functioning of the mapping table is further explained in Appendix C. To compute the network

position measures, we used UCINET for Windows (Borgatti et al., 2002) and Pajek (De Nooy et al., 2005).

*Network closure*

Our measurement of network closure was grounded in the concept of *aggregate constraint*, which builds upon Burt's (2010) network constraint metric. This metric captures information regarding an individual's ego or local network. Where an individual's network is characterized by mutual connections among their associates, their constraint measure is higher; conversely, their constraint measure is lowest when their associates are not interconnected. Aggregating this constraint across all actors yields the measure of aggregate constraint per actor (De Nooy et al., 2005). While aggregate constraint is a relatively infrequent used measure, it holds significant relevance to the study of social capital. It signifies the degree to which a focal actor maintains more advantageous relationships with other actors than those actors do among themselves, thus representing the opposite of structural holes. A high value for aggregate constraint emerges when all actors related to the focal developer are interconnected. Consequently, individuals with high aggregate constraint values possess fewer opportunities for brokering (Burt, 2004). Conversely, lower aggregate constraint values suggest that actors may leverage structural holes to bridge distinct knowledge domains.

The calculation of aggregate constraint involves summing all dyadic constraint values for the focal actor, denoted as $v_i$, which are based on proportional strength (Borgatti et al., 2018). As defined by Burt (1992), proportional strength quantifies the importance of a link between nodes based on the number of connections a node has. Generally, this is achieved by dividing the sum of connections weights between $v_i$ and $v_j$ by the sum of connection weights that $v_i$ has with other nodes. When the focal actor possesses numerous contacts that are isolated from one

another, the value of aggregate constraint tends to approach near 0 (Burt, 1992). Typically ranging from 0 to 1, values for aggregate constraint can exceed 1 in extensive networks (Everett & Borgatti, 2020). In our case, these values range from 0 to 1.28.

*Positional embeddedness*

In network scenarios, the concept of embeddedness is frequently elucidated through references to network centrality such as outdegree or indegree, which essentially quantify the sheer number of connections (e.g., Kratzer et al., 2016; Wasko & Faraj, 2005). However, especially in the context of private-collective innovation, the nature of these connections holds greater significance than their sheer quantity. To capture positional embeddedness, we employed "eigenvector centrality", a metric that has been used to gauge this aspect in prior studies on OSS (e.g., Grewal et al., 2006). Bonacich (2007, p. 555) emphasizes that "*eigenvector centrality is designed to be distinctively different from mere degree centrality when there are some high degree positions connected to many low degree others or some low degree positions are connected to a few high degree others*". This depiction is aptly represented in OSS networks characterized by core-periphery structures. Eigenvector centrality is computed by assigning relative scores to all nodes, where connections to well-connected nodes count greater weight than those to less connected nodes. The metric generates values that span the range from 0 to 1.

*Moderator: Developer groups*

To categorize individual developers into distinct groups, we employed the email postfixes of their email addresses (Dahlander & Wallin, 2006). These groups encompassed "sponsored" individuals and various categories of "non-sponsored" developers, including those associated with the kernel, education, volunteering, and user roles. It is worth noting that individual developers may use multiple email addresses for communication. In such cases, for coding

purposes, we selected the email address from which most messages were sent (refer to Appendix C for details). Due to the immense size of the database, manual coding of the entire dataset was unfeasible, prompting us to adopt the following procedure. Initially, we identified the 600 most active communicators for the years 2000-2008, essentially those contributed 300 or more mailing list posts. These individuals were then categorised based on their email postfixes into the following groups. The first group is sponsored developers with email addresses affiliated with firms – including those officially listed as Linux sponsored firms according to the official Linux kernel report in 2010 (Corbet et al., 2010). Notable contributors in terms of technical contribution included: IBM (17.58%), Intel (10.37%), RedHat (5.15%), HP (4.91%), and Suse (3.33%). The second group is the kernel affiliated developers affiliated with the Linux foundation, identified through the postfix @kernel.org. The third group comprised educational developers linked to educational institutions using postfixes such as "@mit.edu"' or "@harvard.edu". The fourth group, which we labelled volunteer developers identified by email postfixes from prominent email providers such as Yahoo or Gmail. Having manually coded the email postfixes of the 600 most active individuals, we then automated the coding process for the rest of the dataset. However, due to postfix variations beyond the initial 600, not all 11,899 developers could be categorized. Consequently, those with email addresses not matching the predefined 600 postfixes were assigned a *null* value. This procedure successfully categorized around 40% of the mailing list contributors, leaving the remaining 60% labelled as "users". This "user" category encompassed episodic volunteers who are less frequently engaged in Linux kernel development. Table 1 offers an overview of the developer groups, complete with example email postfixes, while Table 2 illustrates the average network centrality for each developer group, including out-degree (i.e. number of peers to which an email was sent) and in-degree

centrality (i.e. number of emails received from unique actors). Additionally, a Scheffé post hoc analysis was conducted to discern group differences.

**Table 1.** Developer groups.

| Developer group | Operational definition | Example |
|---|---|---|
| Sponsored | A developer that works for Linux on behalf of a company | @redhat.com |
| Kernel | A developer that works for the Linux foundation | @kernel.org |
| Education | A developer with an educational Email-address | @harvard.edu |
| Volunteer | A developer that clearly could be defined as neither sponsored, kernel, or education. | @gmx.de / @yahoo.com |
| Users | All developers for which a clear identification was not possible based on the coding scheme for the 600 most active individuals | @addtons.fr |

**Table 2.** Scheffé post hoc analysis across contributing groups.

| | | Out-degree | | In-degree | |
|---|---|---|---|---|---|
| Contributing Groups | n | Mean | SD | Mean | SD |
| Users | 6,535 | 2.70 [a] | 7.25 | 2.91 [a] | 7.07 |
| Volunteer | 3,562 | 10.02 [b] | 42.01 | 10.16 [b] | 38.66 |
| Kernel | 118 | 50.65 [c] | 127.43 | 46.09 [c] | 112.73 |
| Education | 420 | 17.55 [d] | 69.69 | 17.27 [d] | 70.60 |
| Sponsored | 1,264 | 28.11 [e] | 92.01 | 27.16 [e] | 85.89 |

*Note*. Means share the same superscripts (a – e) are not statistically significant different from each other.

4.2.3. Control variables

*Source files*

Being listed as an author of a source file not served only to document individual contributions but also conveyed an individual's competence to the community, potentially

influencing their technical contributions over time. Therefore, we accounted for this impact in our analysis. To assess individual contribution to source files, we cross-referenced our list of individual developers' name or email addresses with the headers of over 11,000 source code files from Linux kernel version 2.6.29.4, as of the beginning of 2009. As a result, the variable "source files" can range from 0 to the total number of source files in this specific kernel version.

*Co-authorship*

In addition to source code authorship, we accounted for the number of *individual* co-authors associated with each individual across all source files. The values for this variable can range from 0 to 27.

*Maintainer dummy*

We used the official list of Linux kernel maintainer to identify our maintainer dummy variable, where 0 = non-maintainer, 1 = maintainer. In the Linux project, the role of a maintainer involves responsibilities such as managing submitted patches, which includes tasks like reviewing and potentially rejecting these submitted patches (Lee & Cole, 2003).

*Experience*

The variable "experience" quantifies the duration in days since a developer's initial post on the Linux mailing list. The range of values spans from 0 (if a developer's first post occurred on the final day of our data collection period) to 3,285. In our regression analysis, we applied the natural logarithm to this value after adding 1 to all observations to prevent calculations involving ln(0).

*Mailing lists*

In OSS communities like Linux, GNOME, or KDE (Dahlander & Wallin, 2006; Germonprez et al., 2017; Kuk, 2006), communication primarily occurs through a central mailing

list, although additional mailing lists are employed for specialized topics or related discussion (Rullani & Haefliger, 2013). For example, GNOME utilized a total of 137 mailing lists (Dahlander & Wallin, 2006). Consequently, we incorporated control for the count of different mailing lists a developer actively participated in between 2006 and 2008, apart from the Linux kernel mailing list. In this context, "active" signifies sending at least one message to a list and receiving at least one message from it. The measure for mailing list involvement ranges from 0 to 111. Similar to the "experience" variable, we applied the logarithm to these values. Table 3 provides an overview of the studied measures and their respective sources.

**Table 3**. Summary of measures and operationalizations.

| Variables | Operational definition | Source |
|---|---|---|
| Technical contribution (Dependent variable) | Sum of accepted lines inserted and deleted in 2009 | Git commits 2009 |
| Network closure | Burt's (2010) aggregate constraint | Mailing-list network data 2006-2008 |
| Positional embeddedness | Eigenvector centrality | Mailing-list network data 2006-2008 |
| Developer group | Affiliations identified by Email-postfix (firm-sponsored developers, kernel developers, education, volunteers, users) | Mailing-list network data 2006-2008 |
| Maintainer dummy | Developers who are officially Linux kernel maintainer, where 0 = non-maintainer, 1 = maintainer | Mailing-list network data 2006-2008 |
| Experience | Number of months since first post | Mailing-list network data 2000-2008 |
| Mailing lists | Number of mailing-lists in addition to the Linux kernel mailing list | Mailing-list network data 2006-2008 |
| Source files | Number of source files in which a focal developer is listed as an author | Linux source code 2.6.29.4 (early 2009) |
| Co-authorship | Number of unique co-authors in source files | Linux source code 2.6.29.4 (early 2009) |

### *4.3. Analytical approach – model specification*

To test our research hypotheses, we used zero-inflated negative binomial (ZINB) regression approach. This was necessary due to the nature of our dependent variable, which was

a count variable with a considerable presence of excess zeros (90.66%). This implies that fewer than 10% of developers had submitted a single CVS commit containing lines of code in 2009. Given the overdispersion of the data (mean = 260.85 and sd = 3869.90), which deviated from Poisson regression assumptions, we employed ZINB. The ZINB method involves estimating the likelihood of technical contribution through a logit model, followed by estimating the negative binomial model itself (Greene, 2011; Kennedy, 2006). In the zero-inflated component of the model, we accounted for experience and developer group dummies to consider the direct influence of the duration of relationships with Linux and potential differences in contribution across different developer groups. To confirm the suitability of ZINB regression over the standard negative binomial approach, we employed both the AIC and the BIC-corrected Vuong test (Desmarais & Harden, 2013). Across our study, all Vuong test statistics were positive and statistically significant, confirming that ZINB model is better than the standard negative binomial model. As a result, we proceeded with the ZINB regression. Furthermore, we addressed the issue of potential lack of independence and unobserved heterogeneity, which could lead to an underestimation standard errors and inflated coefficient significance. To counter this, we employed the robust variance estimator to calculate robust standard errors in our analysis irrespective of whether the errors displayed heteroskedasticity (White, 1980).

**5. Results**

Table 4 shows the descriptive statistics and correlations for the key variables used to test Hypotheses 1 to 5. Notably, all the control and independent variables were significantly correlated with the dependent variable at $p < 0.001$. Certain correlations such as that between network closure, mailing lists and experience, suggest the potential for multicollinearity, which could affect the interpretation of estimates in our models. In the light of this, we ran an Ordinary

Least Squares (OLS) regression to ascertain the Variance Inflation Factors (VIF) values. This step was taken because ZINB does not support VIF computation. The average VIF for our independent variables was 1.76, and the highest value reached approximately 2.74, which was well below the acceptable maximum threshold of 5. Consequently, multicollinearity does not a significant concern across all our models.

**Table 4.** Descriptive statistics and correlations of primary variables used in the study.

| | | Mean | SD | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Technical contribution | 260.85 | 3869.90 | 1.00 | | | | | | |
| 2 | Network Closure | 0.62 | 0.36 | -0.09 | 1.00 | | | | | |
| 3 | Positional Embeddedness | 0.10 | 0.91 | 0.15 | -0.17 | 1.00 | | | | |
| 4 | Source files | 0.99 | 7.98 | 0.14 | -0.15 | 0.43 | 1.00 | | | |
| 5 | Co-authorship | 0.28 | 1.21 | 0.06 | -0.19 | 0.16 | 0.19 | 1.00 | | |
| 6 | Experience | 3.26 | 3.33 | 3.33 | -0.58 | 0.15 | 0.16 | 0.24 | 1.00 | |
| 7 | Mailing lists | 0.53 | 0.78 | 0.78 | -0.58 | 0.37 | 0.31 | 0.31 | 0.63 | 1.00 |

N = 11,899; |correlation coefficient| > 0.08, $p < 0.05$. Dummy variables developer group and maintainer and are not shown.

### *5.1. ZINB regression analysis*

Table 5 reports the estimated coefficients of six ZINB regression models, which were employed to test our research hypotheses. To test our research hypotheses, we added each of our independent variables incrementally. The base model, labelled as “Model control” in Table 5, serves as the foundational model and encompasses all control variables. Model 1 tests the first hypothesis (H1) and incorporates the first independent variable – network closure (measured with aggregate constraint). Building on this foundation, Model 2 introduces an interaction term between network closure and the contributing group dummy variable. Model 3 includes the second independent variable – positional embeddedness (measured using eigenvector centrality) along with its squared term to explore the curvilinear effect. Model 4 extends the analysis by incorporating an interactive term between positional embeddedness and the contributing group

dummy variable, using sponsored developers as the reference group. Model 5 goes on to include the interactive term between positional embeddedness and network closure. Lastly, the last column in Table 5 presents the outcomes of our primary findings based on propensity score matching, which serves as a robustness check for the reliability of our results.

**Table 5**. Zero-inflated negative binomial regression on technical contribution with robustness checks.

| | Model Control | Model 1 H1 | Model 2 H2 | Model 3 H3 | Model 4 H4 | Model 5 H5 | Propensity Score Matching |
|---|---|---|---|---|---|---|---|
| COUNT COMPONENT | | | | | | | |
| Network closure | | -3.805*** | -3.401*** | -3.356*** | -3.649*** | -3.657*** | -2.459*** |
| | | (0.520) | (0.569) | (0.571) | (0.579) | (0.561) | (0.534) |
| Network × Developer^ Users | | | -0.026 | -0.213 | -0.387 | 0.385 | 0.415 |
| | | | (0.762) | (0.760) | (0.843) | (0.838) | (0.799) |
| Volunteers | | | -2.237** | -1.953** | -1.574* | -1.514* | -1.281 |
| | | | (0.716) | (0.719) | (0.741) | (0.736) | (0.717) |
| Kernel | | | -9.652*** | -9.335*** | -10.350*** | -9.978*** | -6.898*** |
| | | | (1.683) | (1.893) | (2.006) | (2.042) | (2.059) |
| Education | | | 2.134 | 1.967 | 1.723 | 2.065 | 2.845 |
| | | | (1.483) | (1.573) | (1.602) | (1.580) | (1.683) |
| Positional Embeddedness | | | | 0.442*** | 0.076* | -0.513* | -0.326 |
| | | | | (0.111) | (0.121) | (0.216) | (0.084) |
| Positional squared term | | | | -0.015*** | -0.006* | -0.004 | -0.001* |
| | | | | (0.003) | (0.003) | (0.004) | (0.004) |
| Positional × Developer^ Users | | | | | 2.646* | 1.800 | 1.630 |
| | | | | | (1.206) | (1.165) | (1.037) |
| Volunteers | | | | | 0.473*** | 0.521** | 0.152* |
| | | | | | (0.146) | (0.171) | (0.152) |
| Kernel | | | | | -0.068 | 0.152 | -0.168 |
| | | | | | (0.220) | (0.252) | (0.226) |
| Education | | | | | -0.131 | -0.027 | -0.002 |
| | | | | | (0.213) | (0.239) | (0.180) |
| Network closure × Positional Embeddedness | | | | | | 8.224*** | 6.061*** |
| | | | | | | (2.357) | (1.857) |
| Developer group^ | -0.221 | -0.212 | -0.332 | -0.243 | -0.758 | -0.610 | -0.515 |
| Users | (0.268) | (0.302) | (0.363) | (0.361) | (0.487) | (0.487) | (0.430) |
| Volunteers | -0.756** | -1.220*** | -0.423 | -0.716* | -1.018** | -0.965** | -0.611 |
| | (0.278) | (0.266) | (0.345) | (0.323) | (0.356) | (0.364) | (0.331) |
| Kernel | 0.202 | -0.244 | 1.583** | 1.459* | 1.789* | 1.686 | 1.573* |
| | (0.463) | (0.532) | (0.588) | (0.684) | (0.766) | (0.775) | (0.781) |
| Education | -1.318** | -0.905 | -2.170** | -2.096* | -1.926* | -2.070 | -2.268* |
| | (0.439) | (0.590) | (0.588) | (0.840) | (0.914) | (0.899) | (0.859) |
| Source files | 0.043*** | 0.052** | 0.052** | 0.062*** | 0.064*** | 0.064*** | 0.061*** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | (0.013) | (0.017) | (0.017) | (0.018) | (0.018) | (0.017) | (0.015) |
| Co-authorship | 0.098 | 0.139 | 0.132 | 0.151 | 0.177* | 0.171 | 0.052 |
| | (0.065) | (0.086) | (0.085) | (0.090) | (0.086) | (0.090) | (0.061) |
| Maintainer dummy | 2.026*** | 2.104*** | 2.225*** | 2.232*** | 2.214*** | 2.241*** | 1.987*** |
| | (0.225) | (0.248) | (0.269) | (0.258) | (0.249) | (0.249) | (0.204) |
| Experience | -0.234** | -0.587*** | -0.623*** | -0.624*** | -0.631*** | -0.643*** | -0.447*** |
| | (0.074) | (0.093) | (0.088) | (0.088) | (0.088) | (0.086) | (0.068) |
| Mailing list | 1.107*** | 0.902*** | 0.803*** | 0.550*** | 0.526*** | 0.608*** | 0.612*** |
| | (0.136) | (0.141) | (0.140) | (0.158) | (0.160) | (0.167) | (0.146) |
| (intercept) | 5.780*** | 6.550*** | 19.549*** | 9.670*** | 9.923*** | 9.786*** | 8.757*** |
| | (0.311) | (0.364) | (0.683) | (0.689) | (0.701) | (0.693) | (0.552) |
| ZERO COMPONENT | | | | | | | |
| Experience | -0.239*** | -0.286*** | -0.289*** | -0.293*** | -0.296*** | -0.296*** | -0.289** |
| | (0.026) | (0.029) | (0.030) | (0.031) | (0.032) | (0.031) | (0.113) |
| Developer group^ | 2.628*** | -2.760*** | 2.693*** | -2.660*** | 2.656*** | 2.672*** | -0.584 |
| Users | (0.308) | (0.227) | (0.220) | (0.220) | (0.217) | (0.216) | (0.334) |
| Volunteers | 2.085*** | 2.170*** | 2.059*** | 2.009 *** | 2.011*** | -2.021*** | -0.700* |
| | (0.298) | (0.230) | (0.222) | (0.222) | (0.220) | (0.219) | (0.354) |
| Kernel | 2.085*** | 2.102*** | 1.613** | 1.616** | 1.598* | 1.633* | -20.656*** |
| | (0.298) | (0.509) | (0.644) | (0.632) | (0.653) | (0.645) | (0.712) |
| Education | 1.989*** | 2.845*** | 2.740*** | 2.706*** | 2.727*** | 2.731*** | -0.540 |
| | (0.491) | (0.376) | (0.362) | (0.365) | (0.365) | (0.256) | (0.973) |
| (intercept) | -0.105 | -0.241 | -0.141 | -0.083 | -0.081 | -0.091 | -0.144 |
| | (0.384) | (0.269) | (0.260) | (0.175) | (0.262) | (0.256) | (0.271) |
| Ln $\alpha$ | 2.909*** | 3.012*** | 2.994*** | 2.981*** | 2.983*** | 2.978*** | 2.184 |
| $\alpha$ | 18.333 | 20.336 | 19.958 | 19.708 | 19.752 | 19.655 | 8.885 |
| Vuong test with AIC correction | 3.95*** | 4.98*** | 5.01*** | 5.07*** | 5.10*** | 5.13*** | |
| Vuong test with BIC correction | 2.89*** | 3.93*** | 3.97*** | 4.04*** | 4.07*** | 4.11*** | |
| LR$\chi$2 | 353.13*** | 442.58*** | 468.78*** | 484.02*** | 493.02*** | 502.43*** | 466.07*** |
| LR chi-square test | | 89.45*** | 26.20*** | 15.24*** | 9.00* | 9.42** | |

*N* = 11,899. Standard errors are enclosed in brackets and are robust to arbitrary heteroscedasticity. * p < 0.05, ** p < 0.01, *** p < 0.001. ^Firm-sponsored developers is reference group for the developer groups (where 1 = users, 2 = volunteers, 3 = kernel developers, and 4 = firm-sponsored developers); Propensity score matched sample *N* = 2,222.

We start with interpreting the control model. We observe that the groups of education ($b$ = –1.318; $p < 0.001$) and volunteer developers ($b$ = –0.756; $p < 0.001$) provide significantly lower levels of technical contribution compared to the sponsored counterparts. However, there are no significant differences observed for kernel and users. Additionally, among the control variables, we find that source files ($b$ = –0.043; $p < 0.001$), maintainer dummy ($b$ = 2.026; $p < 0.001$), experience ($b$ = –0.234; $p < 0.01$), and participation in multiple mailing lists ($b$ = 1.107; $p < 0.001$) exhibit significant influences on technical contribution, leading to fewer source code commits.

For Hypothesis 1, Model 1 of Table 5 indicates a negative and significant coefficient for network closure ($b$ = –3.805; $p < 0.001$). The result supports the Hypothesis 1, which predicted a negative relationship between network closure and technical contribution. Addressing the second hypothesis, which predicted that high network closure would exert a stronger negative impact on technical contribution for firm-sponsored developers in comparison to their non-sponsored counterparts. In line with our expectation, Model 2 of Table 5 indicates firm-sponsored developers were more adversely affected by an increase of network closure when compared to volunteers ($b$ = –2.237; $p < .001$) and kernel developers ($b$ = –9.652; $p < .001$). A visual representation of this interaction is shown in Figure 2. In essence, both the statistical findings and the graphical representation lend support to Hypothesis 2 by highlighting that firm-sponsored developer experienced distinct networking effects compared to volunteers and kernel developers.

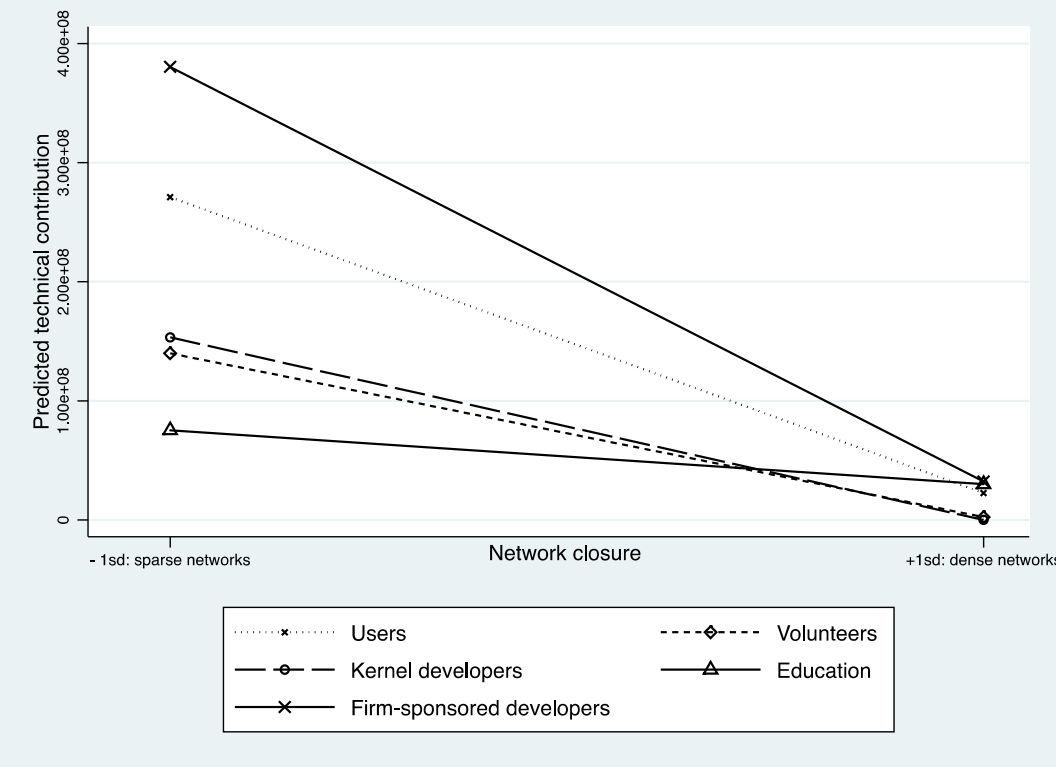


**Figure 2**. Interactive term between network closure and developer groups on technical contribution

Hypothesis 3 posited that a curvilinear relationship between positional embeddedness and technical contribution. Model 3 of Table 5 indicates that the coefficient for the squared term of positional embeddedness is negative and significant ($b$ = –0.015; $p < 0.001$), suggesting a moderate level of positional embeddedness is more conducive to making technical contribution than either extremely high or low levels. This finding aligns with the notion that a balanced degree of positional embeddedness is more optimal for technical contribution. The depiction of this curvilinear relation, charactered by an inverted U-shaped, can be observed in Figure 3. Overall, these results supported Hypothesis 3.

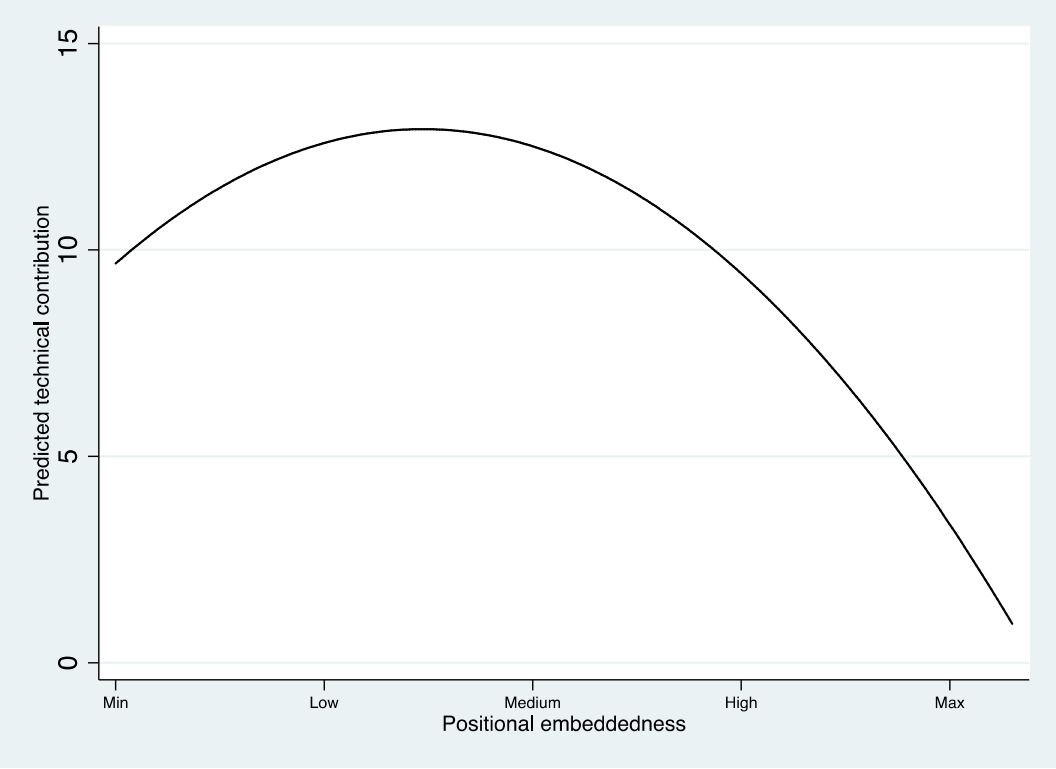


**Figure 3**. Curvilinear relationship between positional embeddedness and technical contribution.

In terms of Hypothesis 4, an analysis of Model 4 from Table 5 reveals interesting insights. It suggests that non-sponsored developers experience a more favourable impact when establishing connections with well-connected nodes. Notably, the coefficients of the interactive terms for users ($b$ = 2.646; $p < .05$) and volunteer developers ($b$ = 0.473; $p < .001$) demonstrate positive and statistically significant associations. This implies that their technical contribution benefits from improved positional embeddedness in contrast to firm-sponsored developers. However, this effect is not applicable to Linux kernel developers, who already possess a strong connection due to their affiliation with the Linux foundation. Figure 4 visually reinforces these significant interaction effects, showcasing that users derive the greatest advantage from high positional embeddedness, while the impact on firm-sponsored developers is less pronounced. Hence, we consider Hypothesis 4 supported.

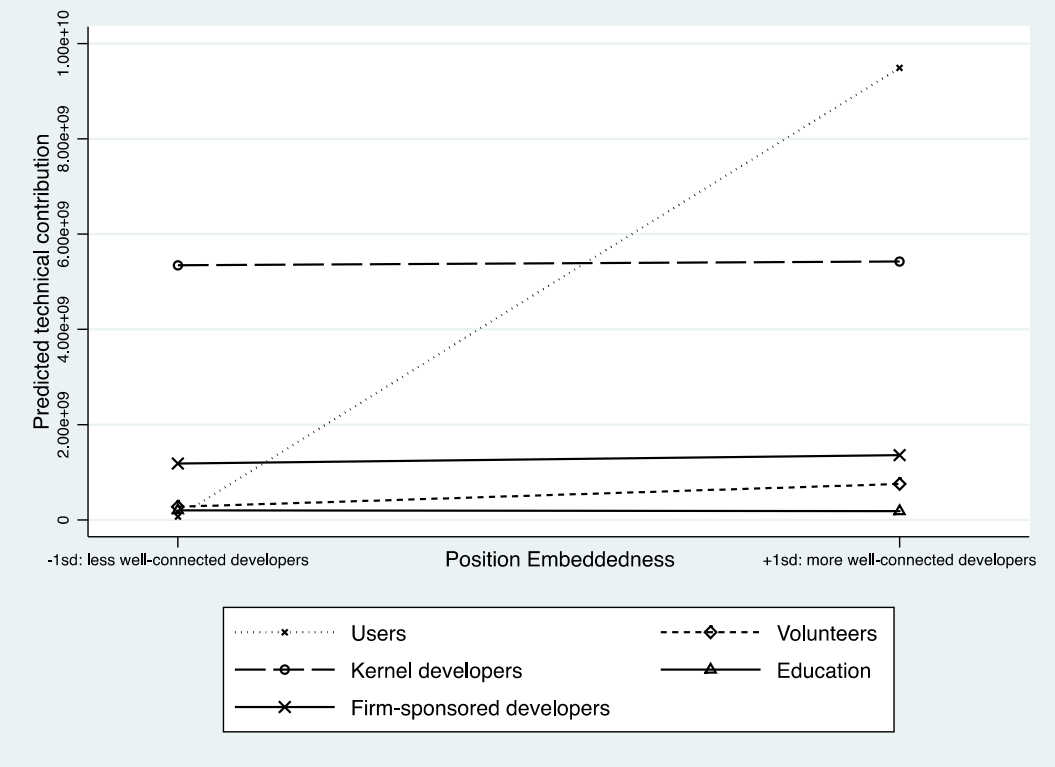


**Figure 4**. Interactive terms between positional embeddedness and developer groups on technical contribution

Lastly, Hypothesis 5 predicted that the joint impact of network closure and positional embeddedness would yield a positive outcome. Upon examining the coefficient of the interaction between positional embeddedness and network closure in Model 5 in Table 5, we find it to be both positive and statistically significant ($b$ = 8.224; $p < .001$). Figure 5 displays the significant interaction effect, illustrating that a combination of high network closure and high positional embeddedness led to a positive influence exerted on technical contribution. These findings provide strong support for Hypothesis 5.

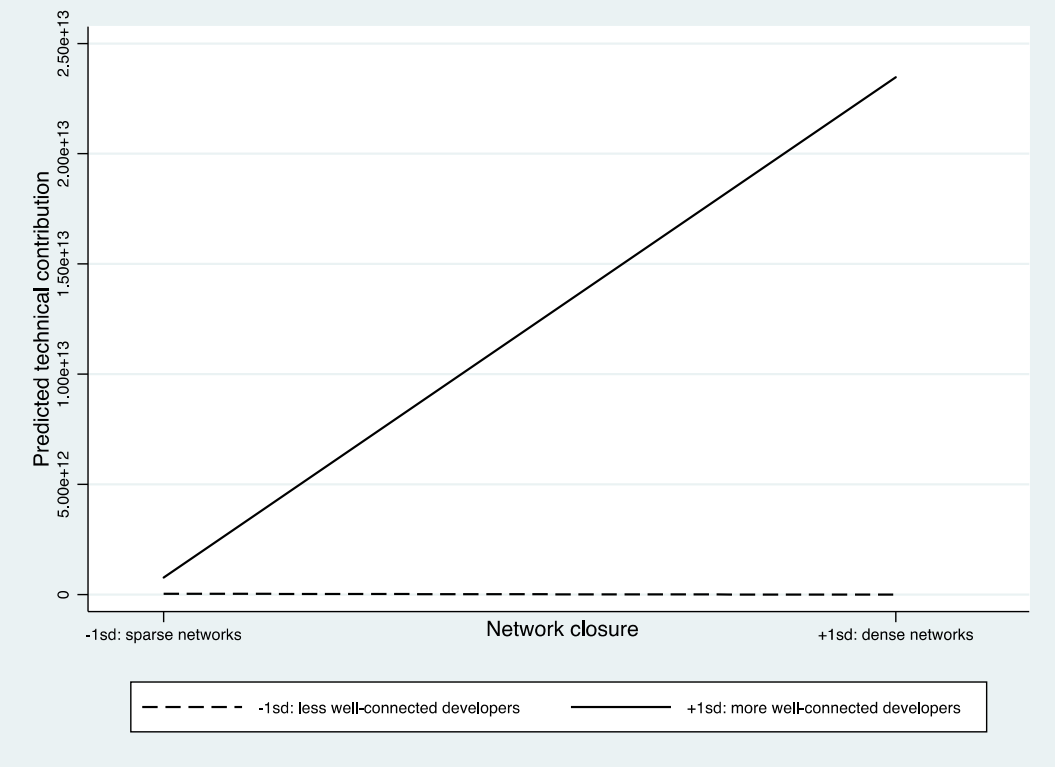


**Figure 5**. Interactive term between network closure and positional embeddedness on technical contribution

*5.2. Robustness Checks*

To ensure the robustness of our findings, we conducted several additional analyses. Firstly, as in common in observational studies (Rosenbaum & Rubin, 1985), we employed propensity score matching to validate the robustness of our main findings (Durward et al., 2020; Lunt, 2014). In this process, we established a treatment dummy, where 1 represented developers who contributed to the Git but were not maintainers, while 0 indicated others, including those who contributed outside their role as Linux kernel maintainers. The treated group was the treated subjects (n = 1,111) falling in the former category, and the untreated group comprised subjects (n = 1,111) belonging to the latter. Next, we used logit regression to calculate the propensity score by regressing the treatment dummy on a distinct set of variables unrelated to the two independent

variables central to our main research hypotheses. These additional variables included all the control variables from our main analysis. We then ran a series of diagnostic tests (see Lunt, 2014) to ensure the accuracy of matched samples, including a graphical inspection of the normal distribution of the propensity scores in both treated and the untreated groups. The Hosmer-Lemeshow Chi-square test (11.78, $p = 0.07$) on initial propensity model indicated a favourable fit of the logit regression to the data. Finally, we used greedy matching technique (using the *gmatch* command in Stata) to compare each treated subject with every untreated subject, facilitating the identification of closely matched pairs based on their propensity scores.

Propensity score matching is commonly used in information systems research (e.g., Durward et al., 2020) and observational studies (Rosenbaum, 1989), particularly valuable for addressing endogeneity concerns (Hamilton & Nickerson, 2003). This technique allowed us to isolate the true impact of the treatment condition. As shown in the second last column "propensity score matching" in Table 5, the coefficients of network closure, positional embeddedness, and their respective interaction terms remained substantively consistent.[6]

In further validation, we reran the analyses by introducing technical contribution in 2008 as an additional control variable. This variable exhibited a statistically significant but minimal effect across all models. While there was a slight reduction in the significance level of Hypothesis 4 (from 95% to 90%), the overall findings remained largely consistent.

## 6. Discussion

Prior research on OSS development and private-collective innovation has primarily taken two approaches: one focused on assessing technical contribution within existing network

[6] As another robustness check, we also used the standard negative binominal analysis to test our research hypotheses. The results remain consistent with the ZINB.

positions, and the other examining the formation of new connections as an outcome of technical contribution. These studies often assume that various developer groups adopted pre-defined roles, limiting our understanding of whether developers are best understood as strategic actors who actively choose projects that align with their unique private interests. This study seeks to address this gap by exploring two distinct forms of networking within private-collective innovation, namely network closure (measured through aggregate constraint) and positional embeddedness (measured using eigenvector centrality).

This study advances the perspective that developers' communication interactions frequently stem from mutually beneficial resource needs, despite differences in ideologies and competing interests. This perspective holds significance as developers of different affiliations often proactively mobilize their own networks of private resources to target specific connections or projects. In a sense, while developers of different affiliations collaborate to attain the shared goal of producing an open source of commercial quality, the underlying mechanisms in driving the realization of their individual private interests tend to vary.

Our primary findings are based in the specific case of Linux kernel development, yet they have the potential for broader application to other OSS projects operating with a private-collective innovation framework (Lee & Baskerville, 2003). They can be summarized as follows (see also Table 5): Firstly, like observations in other network contexts (e.g., Burt, 2017), our research unveils that within private-collective innovation arrangements, high network closure has a negative effect on technical contribution. Consequently, this negatively influences individual private benefits in the form of technical contribution as quantified by source code commits. This finding reaffirms the insights from previous studies within the context of private-collective innovation. Secondly, we highlight that this basic effect is significantly stronger for firm-

sponsored developers. This underscores the significance of our results for managerial considerations. As we will detail in the managerial implications section, firm-sponsored developers need to exercise prudence when forming closed circles. Thirdly, our research reveals that positional embeddedness exerts a curvilinear effect on technical contribution. This finding aligns with prior network studies that were conducted without incorporating the perspective of private-collective innovation. Fourthly, in extending existing knowledge, we demonstrate that the direct impact of positional embeddedness is contingent on the developer group in question. For non-sponsored developers such as users or volunteers, in contrast to sponsored or kernel developers, the association with influential actors holds greater significance. Lastly, we find that positional embeddedness possesses the potential to offset the negative effect of network closure. These findings have implications that we will delve into regarding theory and management considerations.

**Table 5**. Summary of findings and implications

| Hypothesis | Findings | Contributions and Implications |
|---|---|---|
| H1 | Developers who are subject to high network closure (i.e. structural constraints) submit less source code commits (Model 1: b = –3.805, p < .001) | We contribute to existing research, which has mainly concentrated on homogeneous networks to study heterogeneous networks in the private-collective context. Our hypothesis confirms prior network studies of the dampening effect of network closure on technical and innovation outcomes (e.g., Tan et al. 2015). |
| H2 | The association between network closure and source code commits is different for different developer groups. We contrasted firm-sponsored against four other developer groups (i.e. users, | This finding expands network theory by highlighting that within private-collective innovation settings, H1 holds varying significance across different developer groups. While the effect is generally applicable to the |

| | | |
|---|---|---|
| | volunteers, kernel developers, and education). Volunteers (Model 2: $b = –2.237$, $p < .001$) and kernel developers ($b = –9.652$, $p < .001$) are significantly less effected by network closure. Users and education developer are not different from firm-sponsored developers. | entire group of developers, it is notably less pronounced among volunteers and kernel developers when compared to firm-sponsored developers. |
| H3 | Developers who exhibit positional embeddedness (i.e. high eigenvector centrality) submit more source code commits, but only up to a certain point (Model 3, Positional embeddedness: $b = .433$, $p < .001$; positional embeddedness squared: $b = -.015$, $p < .001$) | The finding confirms prior research on the positive effect of positional embeddedness on technical contributions. In addition, we show a curvilinear relationship between positional embeddedness and technical contribution. |
| H4 | The direct association between positional embeddedness and source code commits is different for different developer groups. Users (Model 4: $b = 2.646$, $p < .05$) and volunteers (Model 4: $b = .473$, $p < .001$) exhibit a stronger relation to source code commits than firm-sponsored developers. Kernel and education developer are not different from firm-sponsored developers. | This finding extends network theory by highlighting that within private-collective innovation contexts, H3 varies in its applicability across different developer groups to different degrees. Specifically, the user and volunteer groups derive the most benefit from high positional embeddedness compared to the firm-sponsored developer group. |
| H5 | Network closure and positional embeddedness interact significantly to increase source code commits (Model 5, Network closure × Positional embeddedness: $b = 8.224$, $p < .001$) | This finding furthers network theory by offering insights into the interplay between network closure and positional embeddedness. Notably, positional embeddedness possesses the capability to counteract the negative effect of network closure. |

*6.1. Contributions to theory*

Our study provides several theoretical and empirical contributions. Some of which validate prior research, while others extend existing theoretical frameworks. Firstly, we introduce fresh perspectives into private-collective innovation on both individual and group levels. Prior conceptualisations of private-collective innovation assumed that individual private interests are assimilated within the collective outcome, with private benefits unique to each contributor (Von Hippel & Von Krogh, 2003). We uphold this notion while arguing that technical contributions by various developer groups transcend being solely individual benefits. They hold significance for both the collective entity and the specific interests associated with each group. Consequently, the technical contributions of volunteers, exemplified by source code commits, are likely to sustain individual and community interests. In contrast, firm-sponsored developers' contribution function not only as complements to their sponsoring firms in specific markets but also serves as the bedrock for the project's long-term commercial interests (Alexy & Reitzig, 2013).

Secondly, this study advances the foundations of social network theory and is the first to provide in-depth empirical evidence that reveals how networking varies within private-collective innovation across various developer groups. Due to their heightened interconnectivity, firm-sponsored developers possess a broader social bandwidth (Aral & Van Alstyne, 2011). As opposed to their non-sponsored counterparts, these developers interacting with a pre-existing closed circles are more likely to alleviate the drawbacks of high network closure. Concerning positional embeddedness, non-sponsored developers such as peripheral users or volunteers encounter fewer constraints and reap greater benefits from establishing connections with influential others. In sum, these findings align with Burt's (2000) observation that the effects of network position on outcomes hinge on contextual contingencies. We thereby extend social

network theory within the domain of open source communities by demonstrating the necessity of differentiating between different developer groups when studying network interactions.

This study's final contribution lies in its emphasis on the interplay between network closure and positional embeddedness. The combination of these two facets of network interaction suggests that cultivating strong ties within a well-connected developer network brings about deeper information exchange, fostering trust and consensus among developers. In essence, the findings endorse for a "best of two worlds" strategy for network interaction, signifying the value of forging close connections with other well-connected developers. Yet, when coupled with higher levels of positional embeddedness, the advantages of maintaining close connections outweigh the benefits of acquiring unique knowledge through structural holes. Specifically, well-connected developers pay a crucial role in steering project progression, making connections with them – essentially utilizing borrowed social capital (Burt, 2000) – enables developers to work on modules that have a greater likelihood of garnering positive reviews and eventual inclusion in the primary software release. Our findings hold noteworthy implications for the management of private-collective innovation.

### *6.2. Managerial implications*

In addition to our empirical findings, our theoretical exploration expands the scope of previous managerial understandings regarding unique private benefits. These benefits extend individual levels (which encompass knowledge gains for individual developers) and encompass a spill-over effect onto the organization level. Code contribution, despite being a form of free revealing, reflects unique private benefits for firms, serving to steer the trajectory of an OSS project (Schaarschmidt, 2022). In a similar vein, Alexy, George, and Salter (2013, p. 271) identify free revealing of software code as a firm-level benefit from numerous network

interactions within private-collective innovation. They conclude: “if externals take in the revealed knowledge, because of the cumulative, path-dependent nature of knowledge […], their future knowledge production and spillovers will be of higher value to the focal firm.” Consequently, firms with downstream business models can shape the project’s technical direction by contributing code through their seconded developers.

Moreover, navigating networks within the framework of a private-collective model presents a complex challenge for firms striving to realise private benefits. This task necessitates orchestrating disparate expectations and interests. Given our result of negative consequences of network closure, we suggest developers to engage more actively in brokering. Burt (2000, p. 353) maintains that “structural holes are […] an opportunity to broker the flow of information between people and control the projects that bring together people from opposite sides of the hole.” Thus, firms should enable their seconded developers as intermediaries between the community and the business world.

Lastly, firms can leverage our findings to position developers newly assigned to an open source project. In well-established projects, the core, and consequently, the power to directly influence the project – both through communication and technical contribution – is usually already occupied by in-house developers or those affiliated with rival firms. Our results suggest that beyond merely aiming to move toward the centre, newcomers could instead connect with other influential actors (positional embeddedness) to make their mark.

### *6.3. Limitations*

As with any research, this research is accompanied by certain limitations that warrant acknowledgement. Firstly, the dataset is taken entirely from the Linux kernel project, raising concerns about its generalizability beyond this specific context (Lee & Baskerville, 2003). To

enhance robustness of our findings, future studies should encompasses diverse projects and compare the results with those provided in this study. Secondly, while this study captured three-years of mailing list communication, it assumed networking to remain relatively stable during this period. Although a three-year span appears sufficient given our interest in how firm-sponsored developers establish their network positions, it is crucial to acknowledge that networking dynamics can evolve over time. This prompts the need more longitudinal research that delves into changes in affiliation and alterations in network structure. Thirdly, our focus on the timeframe between 2006 and 2008 was predicated on the surge in company engagement in the Linux domain after 2006. While we don't foresee a reason for data from other time periods to yield divergent outcomes since we were testing theory, there is merit in replicating our findings with data encompassing later market entrants such as mobile app developers. Fourthly, in gauging technical contribution, we used the accepted lines of code inserted and deleted as a proxy. We acknowledge that future research could use alternative proxies for technical contribution to corroborate our findings. Lastly, it is worth noting that sponsoring firms do not haphazardly select modules for contribution. Their choices are often strategic and align with their proprietary interests. Therefore, there is potential for future research refining our findings on more granular levels of technical contribution, further refining our understanding of the relationships.

### *6.4. Conclusion*

Open source software development stands as a prime exemplar of private-collective innovation, where diverse private interests held by various developer groups must harmonise with the collective interest to yield a well-functioning software that serves as both a public good and a top-tier commercial product. Studies in community work have often assumed a shared set

of interests among members, asserting that fundamental principles laid out by social network theory apply uniformly across the board. However, our study uncovers a counterintuitive revelation specific to the context of private-collective innovation. Here, diverse private actors characterized by distinct social structures and the interests linked to them navigate network interactions with varying degrees of impact. This intricate interplay becomes evident: Network closure, when contrasted with volunteers and users, exerts a more detrimental effect on firm-sponsored developers. Similarly, when compared to fellow non-sponsored developers such as users and volunteers, firm-sponsored developers derive comparatively less benefits from positional embeddedness. In essence, this study is the first to explore how diverse interests as reflected by distinct developer groups, translate into nuanced networking behaviours within the realm of private-collective innovation. Moreover, we unveil how these behaviours yield unique private benefits, channelled through technical contribution measured by source code commits.

**References**


Alexy, O., George, A., Salter, A.J. 2013. Cui bono? The selective revealing of knowledge and its implications for innovative activity. *Academy of Management Review* 38(2): 270-291.

Alexy, O., Reitzig, M. 2013. Private–collective innovation, competition, and firms' counterintuitive appropriation strategies. *Research Policy* 42(4): 895-913.

Alexy, O., West, J., Klapper, H., Reitzig, M. 2018. Surrendering Control to Gain Advantage: Reconciling Openness and the Resource-based View of the Firm. *Strategic Management Journal* 39(6): 1704-1727.

Allcott, H., Karlan, D., Möbius, M. M., Rosenblat, T. S., Szeidl, A. 2007. Community size and network closure. *American Economic Review* 97(2): 80-85.

Aral, S. 2016. The future of weak ties. *American Journal of Sociology* 121(6): 1931-1939

Aral, S., Van Alstyne, M. 2011. The diversity-bandwidth trade-off. *American Journal of Sociology* 117(1): 90-171.

Aral, S., Brynjolfsson, E., Van Alstyne, M. 2012. Information, technology, and information worker productivity. *Information Systems Research*, 23(3-part-2): 849-867.

Benkler, Y. 2013. Commons and Growth: The Essential Role of Open Commons in Market Economies. *University of Chicago Law Review* 80(3): 1499-1555.

Bonacich, P. 1987. Power and centrality: A family of measures. *American Journal of Sociology* 92(5): 1170-1182.

Bonaccorsi, A., Giannangeli, S., Rossi, C. 2006. Entry strategies under competing standards: Hybrid business models in the open source software industry. *Management science*, *52*(7), 1085-1098.
Borgatti, S. P., Everett, M. G., Freeman, L. C. 2002. Ucinet for Windows: Software for social network analysis.
Borgatti, S. P., Everett, M. G., Johnson, J. C. 2018. *Analyzing Social Networks*. 2nd ed, Sage Publication, Los Angeles.
Burt, R. S. 1992. *Structural holes*. Cambridge, MA: Harvard University Press
Burt, R. S. 2000. The network structure of social capital. *Research in Organizational Behavior*, 22: 345-423.
Burt, R. S. 2004. Structural holes and good ideas. *American Journal of Sociology* 110(2): 349-399.
Burt, R. S. 2005. *Brokerage and closure*. Oxford University Press: Oxford
Burt, R. S. 2010. *Neighbour networks*. Oxford University Press: Oxford.
Burt, R. S. 2017. Structural holes versus network closure as social capital. *Social Capital*, 31-56.
Chou, S. W., & He, M. Y. 2011. Understanding OSS development in communities: the perspectives of ideology and knowledge sharing. *Behaviour & Information Technology*, *30*(3), 325-337.
Coleman, J. S. 1990. Foundations of Social Theory. Cambridge, MA: Belknap Press.
Corbet, J., Kroah-Hartman, G., McPherson, A. 2010. Linux Kernel Development Report. San Francisco: The Linux Foundation.
Corbet, J., Kroah-Hartman, G., 2017. Linux Kernel Development Report. San Francisco: The Linux Foundation.
Dahlander, L., O'Mahony, S. 2011. Progressing to the Center: Coordinating Project Work. *Organization Science* 22(4): 961-979.
Dahlander, L., Wallin, M. 2006. A Man on the Inside: Unlocking Communities as Complementary Assets. *Research Policy* 35: 1243-1259.
Daniel, S. L., Maruping, L., Cataldo, M., Herbsleb, J. 2018. The impact of ideology misfit on open source software communities and companies. *MIS Quarterly*, 42(4).
Desmarais, B. A., Harden, J. J. 2013. Testing for zero inflation in count models: Bias correction for the Vuong test. *The Stata Journal* 13(4): 810-835.
De Nooy, W., Mrvar, A., Batagelj, V. 2005. *Exploratory social network analysis with Pajek.* Cambridge, UK: Cambridge University Press.
Di Gangi, P., Teigland, R., & Yetis, Z. 2022. How do different stakeholder groups within an open source software project influence the project's development: a case study of OpenSimulator. *Information Technology & People*, (ahead-of-print).
Durward, D., Blohm, I., & Leimeister, J. M. 2020. The nature of crowd work and its effects on individuals' work perception. *Journal of Management Information Systems*, *37*(1), 66-95.
Erickson, K. 2018. Can creative firms thrive without copyright? Value generation and capture from private-collective innovation. *Business Horizons* 61(5): 699-709.
Everett, M. G., Borgatti, S. P. 2020. Unpacking Burt's constraint measure. *Social Networks*, 62: 50-57.
Fang, Y., & Neufeld, D. 2009. Understanding sustained participation in open source software projects. *Journal of Management Information Systems*, *25*(4), 9-50.
Fitzgerald, B. 2006. The transformation of open source software. *MIS Quarterly* 30(3): 587-598.

Fleming, L., Waguespack, D. 2007. Brokerage, boundary spanning, and leadership in open innovation communities. *Organization Science* 18(2): 165-180.
Foss, N., Jeppesen, L. B., Rullani, F. 2020. How context and attention shape behaviors in online communities. A modified garbage can model. *Industrial and Corporate Chang*e, forthcoming
Gächter, S., von Krogh, G., Haefliger, S. 2010. Initiating private-collective innovation: The fragility of knowledge sharing. *Research Policy* 39(7): 893-906.
Gargiulo, M., Ertug, G., Galunic, C. 2009. The two faces of control: Network closure and individual performance among knowledge workers. *Administrative Science Quarterly* 54(2): 299-333.
Germonprez, M., Kendall, J. E., Kendall, K. E., Mathiassen, L., Young, B. Warner, B. 2017. A theory of responsive design: A field study of corporate engagement with open source communities. *Information Systems Research*, 28(1): 64-83.
Granovetter, M. 1985. Economic action and social structure: The problem of embeddedness. *American Journal of Sociology*, 91(3): 481-510.
Greene, W. H. 2011. *Econometric analysis*. Englewood Cliffs, NJ: Prentice Hall.
Grewal, R., Lilien, G., Mallapragada, G. 2006. Location, location, location: How network embeddedness affects project success in open source systems. *Management Science* 52(7): 1043-1056.
Hamilton, B. H., Nickerson, J. A. 2003. Correcting for endogeneity in strategic management research. *Strategic organization* 1(1): 51-78.
He, P., Li, B., Huang, Y. 2012. Applying centrality measures to the behavior analysis of developers in open source software community. In *2012 Second International Conference on Cloud and Green Computing* (pp. 418-423).
Henkel, J. 2006. Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy* 35: 953–969.
Homscheid, D., Kunegis, J., and Schaarschmidt, M. 2015. Private–Collective Innovation and Open Source Software: Longitudinal Insights from Linux Kernel Development. In *Proceedings of the 14th Conference on e-Business, e-Services and e-Society*, Delft, Netherlands, Lecture Notes in Computer Science, LNCS-9373, Open and Big Data Management and Innovation, pp. 299-313.
Howison, J., Wiggins, A., Crowston, K. 2011. Validity issues in the use of social network analysis with digital trace data. *Journal of the Association for Information Systems* 12(12): 767.
Kennedy, P. 2006. *A guide to econometrics*. Cambridge, MA: The MIT Press
Kratzer, J., Lettl, C., Franke, N., Gloor, P. A. 2016. The social network position of lead users. *Journal of Product Innovation Management* 33(2). 201-216.
Kuk, G. 2006. Strategic interaction and knowledge sharing in the KDE developer mailing list. *Management Science* 52(7): 1031-1042.
Laursen, K., & Salter, A. (2006). Open for innovation: the role of openness in explaining innovation performance among UK manufacturing firms. *Strategic Management Journal*, *27*(2), 131-150.
Lee, G. K., Cole, R. E. 2003. From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organization Science* 14(6): 633-649.

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

Linåker, J., Regnell, B., Damian, D. 2019. A Community Strategy Framework–How to obtain Influence on Requirements in Meritocratic Open Source Software Communities?. *Information and Software Technology*. 112: 102-114

Lee, A. S., Baskerville, R. L. 2003. Generalizing generalizability in information systems research. *Information Systems Research*, *14*(3), 221-243.

Lotufo, R., She, S., Berger, T., Czarnecki, K. & Wasowski, A. 2010. Evolution of the Linux Kernel Variability Model, in D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. Bosch & J. Lee, eds, "Software Product Lines: Going beyond", Lecture Notes in Computer Sciences, Vol. 6287, pp. 136-150

Lunt, M. 2014. Propensity analysis in Stata revision: 1.1. Documento disponible en: *URL: http://personalpages.manchester.ac.uk/staff/mark.lunt/propensity_guide. pdf*.

Maruping, L. M., Daniel, S. L., Cataldo, M. 2019. Developer centrality and the impact of value congruence and incongruence on commitment and code contribution activity in open source software communities. *MIS Quarterly*, 43(3): 951-976.

Masmoudi, H., Den Besten, M., De Loupy, C., Dalle, J. M. 2009. *Peeling the onion*. In IFIP International Conference on Open Source Systems (pp. 284-297). Springer, Berlin, Heidelberg.

Owen-Smith, J., Powell, W.W. 2004 Knowledge networks as channels and conduits: the effects of spillovers in the Boston biotechnology community. *Organization Science*,15: 5–21.

O'Mahony, S., Bechky, B. 2008. Boundary organizations: Enabling collaboration between unexpected allies. *Administrative Science Quarterly* 53: 422-459.

Packard, G., Aribarg, A., Eliashberg, J., Foutz, N. Z. 2016. The role of network embeddedness in film success. *International Journal of Research in Marketing*, 33(2): 328-342.

Polidoro Jr, F., Ahuja, G., Mitchell, W. 2011. When the social structure overshadows competitive incentives: The effects of network embeddedness on joint venture dissolution. *Academy of Management Journal*, 54(1): 203-223.

Provan, K. G., Lemaire, R. H. 2015. Positional embeddedness in a community source software development project network: the importance of relationship intensity. *R&D Management*, 45(5): 440-457.

Qu, W. G., Yang, Z., & Wang, Z. 2011. Multi-level framework of open source software adoption. *Journal of Business Research*, *64*(9), 997-1003.

Ramaswamy, V., & Ozcan, K. 2018. What is co-creation? An interactional creation framework and its implications for value creation. *Journal of Business Research*, *84*, 196-205.

Resch, C., Kock, A. 2020. The influence of information depth and information breadth on brokers' idea newness in online maker communities. *Research Policy*, 104142.

Rosenbaum, P. R. 1989. Optimal matching for observational studies. *Journal of the American Statistical Association* 84(408): 1024-1032.

Rosenbaum, P. R., Rubin, D. B. 1985. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician* 39(1): 33-38.

Riehle, D., Riemer, P., Kolassa, C., Schmidt, M. 2014. Paid vs. volunteer work in open source. *47th Hawaii International Conference on System Sciences* (pp. 3286-3295). IEEE.

Rullani, F., Haefliger, S. 2013. The periphery on stage: The intra-organizational dynamics in online communities of creation. *Research Policy* 42(4): 941-953.
Runfola, A., Perna, A., Baraldi, E., Gregori, G. L. 2017. The use of qualitative case studies in top business and management journals: A quantitative analysis of recent patterns. *European Management Journal*, *35*(1): 116-127.
Santos, C., Kuk, G., Kon, F., Pearson, J. 2013. The attraction of contributors in free and open source software projects. *The Journal of Strategic Information Systems*, 22(1): 26-45.
Schaarschmidt, M. 2012. *Firms in open source software development: Managing innovation beyond firm boundaries*. SpringerGabler, Wiesbaden.
Schaarschmidt, M., Walsh, G., von Kortzfleisch, H. F. 2015. How do firms influence open source software communities? A framework and empirical analysis of different governance modes. *Information and Organization* 25(2): 99-114.
Schaarschmidt, M. 2023. Innovating beyond firm boundaries: resource deployment control in open source software development. *Information Technology & People*, 36(4): 1645-1668.
Singh, P. V., & Tan, Y. 2010. Developer heterogeneity and formation of communication networks in open source software projects. *Journal of Management Information Systems*, *27*(3), 179-210.
Singh, P. V., Tan, Y., Mookerjee, V. 2011. Network effects: The influence of structural capital on open source project success. *MIS Quarterly* 35(4): 813-829.
Shah, S. K. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52(7): 1000-1014.
Shaikh, M., Levina, N. 2019. Selecting an open innovation community as an alliance partner: Looking for healthy communities and ecosystems. *Research Policy*, 48(8): 103766.
Spaeth, S., von Krogh, G., He, F. 2015. Research note—Perceived firm attributes and intrinsic motivation in sponsored open source software projects. *Information Systems Research* 26(1): 224-237.
Tan, J., Zhang, H., Wang, L. 2015. Network Closure or Structural Hole? The Conditioning Effects of Network–Level Social Capital on Innovation Performance. *Entrepreneurship Theory and Practice* 39(5): 1189-1212.
Tang, T. Y., Fang, E. E., Qualls, W. J. (2020). More Is Not Necessarily Better: An Absorptive Capacity Perspective on Network Effects in Open Source Software Development Communities. *MIS Quarterly*, *44*(4).
Teigland, R., Di Gangi, P. M., Flåten, B. T., Giovacchini, E., Pastorino, N. 2014. Balancing on a tightrope: Managing the boundaries of a firm-sponsored OSS community and its impact on innovation and absorptive capacity. *Information and Organization* 24(1): 25-47.
Temizkan, O., Kumar, R. L. 2015. Exploitation and Exploration Networks in Open Source Software Development: An Artifact-Level Analysis. *Journal of Management Information Systems* 32(1): 116-150
Toral, S. L., Martínez-Torres, M. R., Barrero, F. (2010). Analysis of virtual communities supporting OSS projects using social network analysis. *Information and Software Technology*, *52*(3): 296-303.
Tortoriello, M., McEvily, B., Krackhardt, D. 2015. Being a catalyst of innovation: The role of knowledge diversity and network closure. *Organization Science* 26(2): 423-438.
Von Hippel, E., von Krogh, G. 2003. Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science* 14: 209-225.

Von Hippel, E., von Krogh, G. 2006. Free revealing and the private-collective model for innovation incentives. *R&D Management* 36(3): 295–306.
Von Krogh, G., Haefliger, S., Spaeth, S., Wallin, M. W. 2012. Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly*, 649-676.
Walter, J., Lechner, C., & Kellermanns, F. W. 2007. Knowledge transfer between and within alliance partners: Private versus collective benefits of social capital. *Journal of Business Research*, *60*(7), 698-710.
White, H. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4): 817-838.
Ye, Y. & Kishida, K. (2003). Toward an understanding of the motivation of open source software developers. *25th International Conference on Software Engineering*, p. 419-429.
Zukin, S., DiMaggio, P.J. 1990. *Structures and Capital: The Social Organization of the Economy*. Cambridge University Press, New York.

**Appendix to:**
**All of the same breed? A networking perspective of private-collective innovation**

**Appendix B.** Additional information concerning the data cleaning process.

To arrive at a sufficient data quality and to calculate network measures only for persons and not machines or bots, we used the following procedure. After we had finished data collection, the entire data set contained 1.33 Mio messages from about 107,000 different Email addresses. In a first step, we deleted all messages that were unconnected to Linux development. These messages are identifiable because their subject is “hi”, “hello”, “no subject”, “subscribe”, “error”, “help”, “question” or “test”. By deleting these messages, we arrived at approximately 1.25 Mio messages. In a mapping table, we mapped different Email-addresses to one identity. According to this table, the data set contained about 90,000 sender identities (including mail-bots). Deleting those who are connected to the messages mentioned before (e.g., “hi”) together with identities that represent computers (e.g., mailer-daemon, postmaster, etc.) led to a set of about 60,000 developers. Finally, we only included in our analysis developers that posted between 2006-01-01 and 2008-12-31. The final sample consists of 11,899 developers.

**Appendix C.** Functioning of the mapping table to map developer identities and Email-addresses.

The data was collected based on messages sent to the Linux kernel mailing list. Message headers contained both the Email-address that was used and an associated sender name in the form of ‘first name’ ‘last name’. In many cases, the same Email-address was associated with different names and vice versa. In case we would use message data only, we would have noise in the data due to developers that were counted more than once and with potentially different affiliations. To this end, we developed a mapping table (see also Appendix B) together with an algorithm. The basic idea of the mapping table is to map multiple entries of the same person to a single identity. Thus, the algorithm checked for all names and for all Email-addresses, whether a name was associated with more than one Email and whether an Email was associated with more than one name. In the few cases where one Email-address was associated with more than one name, we used the name that was used last. In cases one name was associated with more than one Email-address, we mapped the name to the Email address with which the person had sent most messages. We note that with this approach we cannot detect very seldom incidents such as two unique developers that both have the same name or an Email-address that has been passed from one developer to another.

**Appendix A.** Non-exhaustive list of relevant research on network analysis of open source software development

| Authors | Analysis | Theoretical Framework | Role of centrality | Centrality measures used | Outcomes considered | Main finding in relation to centrality | Actor types considered? |
|---|---|---|---|---|---|---|---|
| Chou & He (2011) | Survey, individual level, N=114 OSS developers | Social capital | Independent variable | Questionnaire items | Expertise integration Task completion | Members with higher levels of network centrality have a positive effect on expertise integration of OSS teams | No |
| Dahlander & Wallin (2006) | Archival data, N=1,659 individuals, GNOME community | Appropriability regimes | Dependent variable | Indegree, Outdegree, Prestige | Outdegree, Indegree, Prestige | Firm sponsored individuals interact with more individuals than interact with them, and also they seek to interact with central individuals in the community. | Yes, paid and unpaid an independent variable |
| Dahlander & O'Mahony (2011) | Archival data, N=24,694 individuals and N=74,249 individual years, GNOME community | Lateral authority | Independent variable (control variable) | Degree centrality | Project membership, Board director | Technical contributions are initially important; Coordination work is more critical at a subsequent stage to gain lateral authority. | Yes, affiliation as control |
| Grewal et al. (2006) | Archival data, N=490 developers, Perl foundry | Signaling theory | Independent variable | Degree centrality, Betweenness centrality, Eigenvector centrality | Number of commits, number of downloads (for projects) | Network embeddedness has strong and significant effects on both technical and commercial success, but these effects depend on certain regimes. | No |
| He et al. (2012) | Archival data, N=5363 to 7065 nodes, Flossmole | No theoretical anchor, descriptive | Aggregate centrality measures per year | Closeness centrality, Betweenness | - | New developers tend to collaborate more likely with existing members | No |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | projects hosted on sourceforge | | | centrality, Degree centrality | | as opposed to other newcomers, as existing members often have more important position and richer experiences. | |
| Maruping et al. (2019) | Archival data and survey, N=410 individuals, GNOME community | Uncertainty reduction theory, Person-environment fit | Moderator and independent variable | Degree centrality | Code contribution activity | Results suggest that developer centrality moderates the impact of congruence and incongruence in OSS values on commitment. | Yes, paid and unpaid |
| *This study* | *Network analysis, authorship, and code contribution of N=11,899 Linux developers* | *Social network theory* | *Independent variable* | *Aggregate constraint; Eigenvector centrality* | *Code contribution activity* | *Network closure has a negative effect on code contribution, but this effect is contingent on developer types.* | *Yes, paid and unpaid* |