Bachelor's Thesis

EXPLORATION OF PREFERENTIAL ATTACHMENT IN OPEN-SOURCE SOFTWARE PROJECTS

SVEN FEHLMANN

June 15, 2021

Advisor:
Thomas Bock Chair of Software Engineering

Examiners:
Prof. Dr. Sven Apel Chair of Software Engineering
Prof. Dr. Sebastian Hack Compiler Design Lab

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

UNIVERSITÄT
DES
SAARLANDES

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,_____ _____

(Datum/Date) (Unterschrift/Signature)

ABSTRACT

Open-source software is becoming more and more important in multiple areas of computer usage like productivity software, i.e. office tools or mailing software, or embedded systems. Be it for operating systems like linux distributions, free alternatives to commercial software or even security relevant software.

Using data about code commits and mailing-lists from open-source software projects, we create author-based networks to further analyse network evolution, i.e. how new connections in the network are created, based on data from real open-source software projects. More precisely, we analyse three open-source software projects regarding the preferential attachment model, i.e. whether or not their developer networks evolve in a way, that those developers who are already highly connected, acquire most new links in the network. This is done by splitting the data into subnetworks, also called timeframes. Using these subnetworks, we find out the time periods when a projects developer network evolved based on preferential attachment and discuss the reasons behind the preferential attachment evolution for each project. This is done in order to get a better understanding of how developers connect to each other in an open-source community.

Busybox, OpenSSL and QEMU were chosen as the projects used for analysis, because they all are well known, use mailing-lists for communication, have a long history of data that can be analysed and have a non-negligible number of contributors (the lowest number being over 3000 contributors for Busybox).

For the two smaller projects, Busybox and OpenSSL, we found out that on mailing lists non-new developers tend to connect to other non-new developers that already are highly connected, most prominently during times without noticebly large mail traffic on the mailing-list threads.

For QEMU, having nearly ten times more new connections between developers than the Busybox or QEMU, no such correlation could be observed.

Additionally, the chosen length of the timeframes partially does have impact on whether or not timeframes are considered to evolve based on preferential attachment. However, no generalization for which timeframe length has which effect on the network could be made, because results between the projects vary.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

1 INTRODUCTION

Open-source software is becoming more and more important in multiple areas of computer usage like science or even entertainment. Be it for operating systems like linux distributions, free alternatives to commercial software like Blender or Gimp, or libraries and frameworks for other software projects to build upon, e.g. React by Facebook or Angular by Google. Major software companies, as mentioned before, release their toolkits under open-source licenses, so people all over the world can use them while also contributing to them with feature requests, issue reports and code commits. There are multiple reasons why a company opensources their software, e.g. *a*) selling services complimentary to their software or *b*) reducing cost by outsourcing developer work to a community [1].

Also, open-source software projects heavily rely on their community. Most projects do not have different layers of management, where an employee has to do what the boss says. While terms of governance in different open-source communities might differ from each other, a project should follow some principles in order to stay healthy [17]. One of these principles is the autonomous participation principle, in which a potential contributer should be allowed to contribute on their own terms, otherwise an open-source community will not be able to grow and survive [17].

When someone wants to join an open-source community, this newcomer must first learn about contribution guidelines if there are any and, in case he plans to submit code, understand the code base. There are several ways, this can be achieved, e.g. reading of available documents or asking questions on mailing lists. The second option would already lead to active involvment in the community. When someone answers the question or joins the conversation, connections from the newcomer to the community network are established. These connections are not limited to mail conversation though, as code contributions that are related to other peoples contributions also create connections in the network. While reasons for joining an open-source community are already well studied [1], the way someone joins an open-source community did not receive so much attention in the past.

1.1 GOAL OF THIS THESIS

All developers who join an open source software project sooner or later will connect with other people in that project, but there is no definite answer on how they connect to the network. Are there any developers who are more preferred than others or do new connections follow a more random approach? Multiple types of real-life networks seem to evolve based on the preferential attachment hypothesis [13], but does this also map onto open-source software projects?

While a macroscopic analysis of different network types, including collaboration networks, was already conducted by Barabási and Jeong [13], this thesis aims to cover the following research questions on specific open-source software projects as part of the collaboration-network area.

RQ1. How do new authors connect to a developer network?

Do newcomers connect to other newcomers or do they try to connect to already active authors? An onboarding technique for open-source projects to acquire new developers is core developers mentoring newcomers[11]. For newcomers to receive mentoring, they therefore should first connect to already active developers who can either mentor them or forward them to core developers who can offer them mentoring, instead of connecting to other newcomers, which could result in a slowdown of integration because of potential missing mentoring.

RQ2. Do these projects evolve based on preferential attachment?

Do authors, who are already highly connected, receive more new connections than those, who are not that highly connected? A developer who is already highly connected, most likely can be considered a core developer. When a developer network evolves based on preferential attachment, most new network connections are acquired by such core developers, which would mean that most developers would preferably contact core developers, enabling better mentoring opportunities[11].

RQ3. For those projects, that evolve based on preferential attachment: Is there a relation between preferential attachment and the amount of activity in the project?

We will examine moments, where the networks evolved based on preferential attachment and find out whether or not there were more or less mails or commits than during those moments, where the networks did not evolve based on preferential attachment. If there is a correlation between both activity and preferential attachment, one could already estimate whether or not a project could have evolved based on the preferential attchment model, based on the projects activity, before having to conduct a full analysis for preferential attachment.

RQ4. What impact do different timeframe lengths have on analysis results?

For better understanding of the evolution of a whole network, the network can be split into timeframes of the same length. This length however can be chosen arbitrarily, with no information about the impact of different timeframe lengths on the results. Analysing these timeframe lengths can help future preferential attachment studies to choose a fitting length for splitting networks.

Besides timeframe lengths, do different splitting techniques, have an impact on analysis? The reason for using either splitting type is simple, being wanting to conduct a microscopic or macroscopic analysis on the projects network, whereas there is no comparison between the results from the two used splitting types regarding preferential attachment.

These questions will be answered based on observation and evaluation of the data from mailing-lists and commits from various open-source software projects, namely Busybox, OpenSSL and QEMU.

1.2 OVERVIEW

In Chapter 2, we will explain networks as a core concept for this work as well as important network properties. Additional to that, the network building and network splitting that is used in this work will be described. Besides that, we will clarify network growth models, preferential attachment in particular, as well as the scale-freeness property.

Chapter 3 will cover the steps, that were needed to gain the results, needed for this thesis. First, lots of data about differently sized software projects was collected using the Codeface framework by Siemens[1]. The collected data mainly consists of mail headers and code-commit metadata. Everything in that dataset is timestamped, so the first step will consist of building networks from the given data and split those networks based on a fixed timeframe length. After that, activity and connectivity metrics are computed based on the networks, so these metrics then can be used to plot and analyse the networks any further. Using plottet data, we can identify interesting and abnormal parts in the timeline of the respective software project and try to map it to events in the projects timeline. While we can already try to identify patterns indicating preferential attachment, based on the visual look of these plots, we will use power-law fitting in order to base the findings on solid research.

Chapter 4 will contain results, a discussion part as well the discussion of potential threats to validity. You can find related work to this thesis in Chapter 5. Chapter 6 will cover the conclusion of this thesis as well as possible future work on this topic. Visualizations for the results can be found in Appendix A.

1 See: https://github.com/siemens/codeface

2

BACKGROUND

This chapter will cover the basics that are needed to understand the work presented in this thesis. First, networks, network building and the process of splitting them into smaller sub-networks will be explained, because network analysis will not take care of the whole network but needs to be more fine-grained. Then we will discover different network-growth models including preferential attachment, which will be explained in more detail.

2.1 NETWORK THEORY

Networks are the core concept utilized in this work. In simple terms a network is a graph. It consists of edges and vertices. In mathematical terms we write $G = (V, E)$. Each edge $e \in E$ connects two vertices $v_1, v_2 \in V$, which is written as $e = \langle v_1, v_2 \rangle$. Each vertex also has its own vertex degree. We declare $V(N)$ as the function returning all vertices (or nodes) for the network N and $E(N)$ as the function returning all edges for the network N.

VERTEX DEGREE The vertex degree of a vertex v is the number of edges that are connected to the given vertex v and is written as $\delta(v)$[21]. It is important to observe that the degree of a node can also be affected by network simplification[1].

DEGREE DISTRIBUTION The degree distribution is a probabilistic distribution describing the vertex degrees of all network nodes and is the basis for all network-growth models. It is written as $\Pi(k)$, where k is the degree of the nodes $\Pi(k)$ is describing [13]. The form of this distribution gives us more insight in the way a network, we want to analyse, grows. This analysis is done by fitting the distribution to a growth model.

NETWORK SIMPLIFICATION A simplified network removes redundant as well as self referencing edges, so there is no edge $e = \langle v, v \rangle$ and
$\forall e_1, e_2 \in E; e_1 = \langle v_1, v_2 \rangle \wedge e_2 = \langle v_1, v_2 \rangle \Rightarrow e_1 = e_2$[10].
Simplifying a network can cause a difference in vertex degrees, since nodes, that receive n edges from m nodes, where $n > m$ in a non-simplified network, will only receive $x \leq m$ edges in a. x is less or equal to m, because if there already exists an edge between two nodes, no new edge will be created between them. This work uses simplified networks for analysis, because we are only interested in single connections from one person to another. Otherwise, also the quantity of interaction between two nodes would increase the vertex degree and lead to the assumption, that more people connected with one person than they did in reality.

DIRECTED NETWORKS Edges in a network can be directed, which means that a node has outgoing and incoming edges [6]. It also adds more information to the vertex degree as the overall vertex degree now consists of an incoming vertex degree and an outgoing vertex

1 See paragraph 2.1

degree. The analysis in this thesis uses directed networks, so future work can also analyse directions of interactions, e.g. who started a conversation on a mailing list or who did answer another person. This thesis however does not directly make use of directions.

GROWTH MODEL A growth-model is a form of a vertex degree distribution $\Pi(k)$. There are multiple growth models, e.g. Erdős–Rényi model, Watts-Strogatz model and the preferential attachment model. Latter will be explained in the next subsection. The Erdős–Rényi model is meant to describe a fully random network and has two variants. The first variant, given n for the amount of nodes and M for the amount of edges, returns a random network out of the set of networks that have n nodes and M edges [4]. The second variant is based on the probability p, two nodes should connect. For the network generation based on this variant, a random-number generator will decide for each pair of nodes whether or not, an edge will be created between them. The Erdős–Rényi model will be used in this thesis in order to generate randomized networks based on real network configurations, i.e. the amount of nodes n and edges m of a real network will be inserted into the model. This variant is easier to use in this setup than the second variant, therefore the second variant will not be used in this thesis. The Watts-Strogatz model on the other hand creates networks fulfilling the small-worldness property, i.e. clustering in a way, that most nodes, even though they are not directly connected to each other, can be reached with a short path [12]. This model is only described in this thesis in order to give an overview about how different growth models can be. The only two models that will be used are the Erdős–Rényi model as well as the preferential attachment model.

PREFERENTIAL ATTACHMENT Preferential attachment is a growth-model that describes the probability $\Pi(k)$ of network vertices with vertex degree k receiving a new edge to another vertice. In a network that follows the preferential attachment model, vertices that already have a high vertex degree have a higher chance of receiving a new edge to another node than those that do not have a high vertex degree, so $\Pi(k)$ is a monotonically increasing function [13]. Networks in this work will be analysed for preferential attachment properties, because it is one of the more precise network models for real-world networks [13]. The time evolution for preferential attachment is given by the following differential equation:

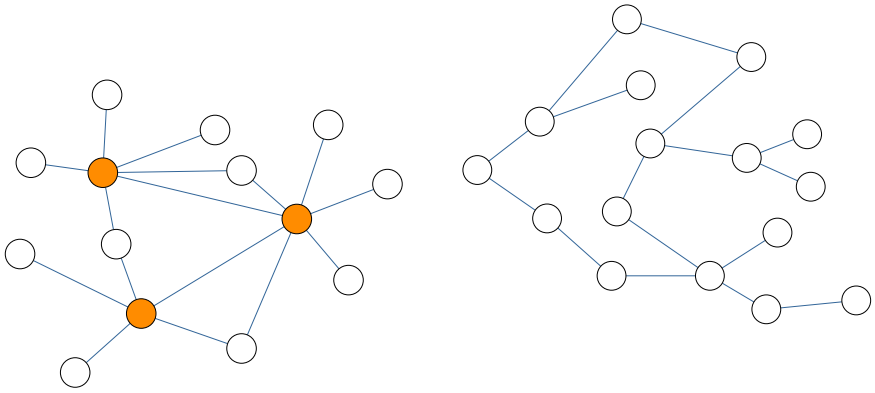
$$\frac{dk_i}{dt} = m\Pi(k_i), \qquad \Pi(k_i) = \frac{k_i^\alpha}{\sum_j k_j^\alpha} = C(t)k_i^\alpha, \tag{2.1}$$

where m is a constant[13]. As a simplification, one can say that $\Pi(k)$ returns the rate at which a node with the vertex degree k acquires a new edge [13].

The probability $\Pi(k)$ is assumed to follow a power law with the scaling exponent $\alpha > 0$. Depending on α, the preferential attachment can be a) linear for $\alpha = 1$, b) sub-linear for $\alpha < 1$ and c) super-linear for $alpha > 1$[3]. For $\alpha = 1$, the degree distribution can be reduced to the scale-free model[13] (see below).

SCALE-FREENESS A scale-free network can be distinguished from non scale-free networks by the presence of highly connected nodes[15]. Its degree distribution can be fit to a power-law distribution [2]. When a network evolves based on preferential attachment, it returns a

scale-free network[15]. Scale-freeness only takes into account the degree distribution of the network[2] and is not determined based on network evolution unlike preferential attachment.



(a) (b)

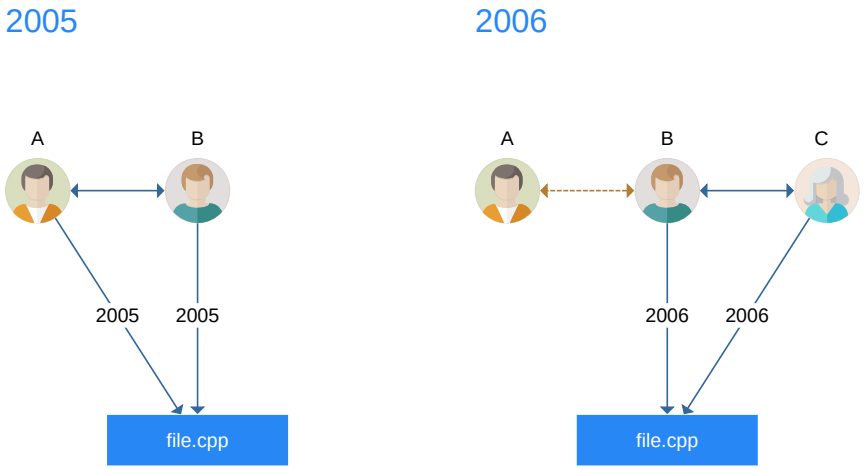
Figure 2.1: Examples for different splitting approaches: In (a), you can see an example for a scale-free network. The highlighted have a noticeable higher vertex-degree (5-6) than the non-highlighted nodes (1-2); (b) shows an example for a non-scale-free network. There is no node with a vertex-degree highly exeeding other nodes vertex degrees (1-3).

2.2 NETWORK APPLICATION

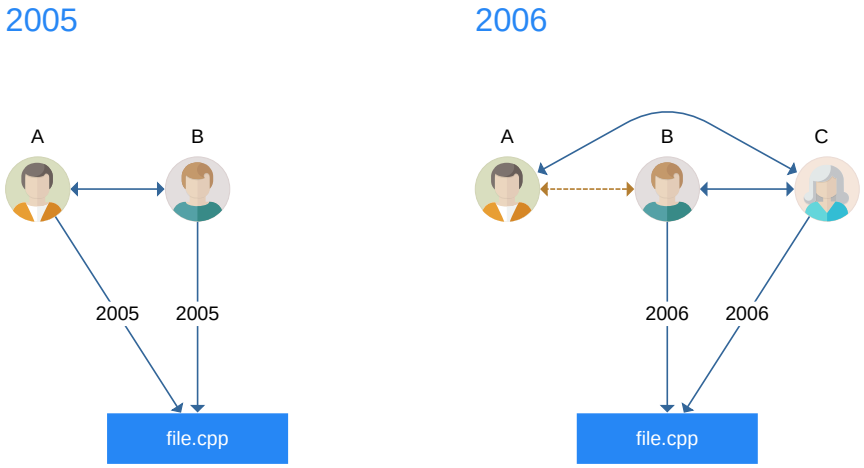
NETWORK BUILDING Building a collaboration network for an open-source software project can be done by making use of *a*) email and *b*) commit data. For email data, we look at whole conversations on the mailing lists. Each email for a topic creates an edge between the sender and everyone else, who participated on this topic. For commit data, we use a file based approach where an edge is created for every two authors who made a commit on the same file. When such a network is built, we only see the final network with all nodes and edges that were formed by the time the data was collected. In order to see an evolution of the network, we must create some sort of snapshots of the network at different dates.

NETWORK SPLITTING This work focusses on time-split networks, which are split into timeframes of the same length. Each of these time-split networks is a subnetwork of the whole network. Splitting can either be done *a*) Node-Based Figure 2.2a and *b*) Edge-Based Figure 2.2b. The Node-Based approach only looks at actions of the current time frame to determine edges between vertices, while the Edge-Based approach also looks at the history, so edges are also created for actions that were conducted in the past, e.g. author A edited file *main.c* in a timeframe older than the current one. Author B then edits the same file in the current timeframe. This creates an edge in the Edge-Based approach, even though the first edit was not done in the current time frame.

Depending on the splitting method, a more microscopic analysis (in case of node-based splitting), i.e. timeframes are isolated from the rest of the network, or a more macroscopic analysis (in case of edge-based splitting), i.e. edges for each timeframe can form based on actions from past timeframes, can be conducted. It is to be noted that the choice of splitting method can also have an impact on the resulting degree distribution, since some people may have lots of incoming edges in a historical view but they do not receive that much edges in the currently viewed timeframe. Additionally to the splitting type, the network can be cut using the sliding-window approach. With this approach, for every neighbouring pair of timeframes, another timeframe is cut, having its starting date in the center of the first timeframes timeline and its end date in the center of the second timeframe. When timeframes are cut at poorly chosen dates, e.g. without sliding-windows, when a timeframe gets cut in t he midst of a high rate of new link acquisition, these new links would then be split between two timeframes. With the sliding-window approach, another timeframe including all of these new edges would then be created, revealing that high-activity situation, that otherwise would have stayed hidden.



(a)



(b)

Figure 2.2: Examples for different splitting approaches. The timeframes are chosen to be one year long, starting at the first of january. (a) describes a node-based approach: When developers A and B both modify file.cpp in the year 2005, they create an edge to each other in the developer network. When in the year 2006, developer B and C both edit file.cpp, they also create an edge to each other. no edge is created between developer A and C, because developer A did not edit file.cpp in the same timeframe as developer C; (b) describes an edge-based approach: When developers A and B modify file.cpp in 2005, they again construct an edge to each other When in year 2006 developers B and C modify file.cpp, they again create an edge towards each other, but this time, developer C also creates an edge to developer A, because developer A modified file.cpp in the past.

3 APPROACH

In this chapter, the approaches that were used to answer the research questions from Chapter 1 will be described in detail. We will go over techniques that help finding out how newcomers join a network and to decide whether or not a network can be mapped on the preferential attachment model. If a networks evolution can be partially mapped onto the preferential attachment model, i.e. we find timeframes in the network that evolve based on preferential attachment, we can go a step further and try to find out if there is a correlation between those time periods where it can be mapped onto the model and time periods of high or low activity on the mailing list and code contributions, i.e. whether or not there are more or less mails or code commits than usual. In a last step, we will compare different network configurations, i.e. towards each other, meaning we will combine different timeframe lengths and splitting types, in order to find out similarities between them.

In a last step, we will compare results from different configurations, i.e. combinations of different timeframe lengths and splitting types, and try to find out how different configurations affect the preferential attachment analysis.

3.1 PROJECT OVERVIEW

The three software projects that are analysed in this thesis, namely *a*) Busybox[1], *b*) OpenSSL[2] and *c*) QEMU[3], are all open-source projects. They have in common, that their main communication medium are mailing-lists[4]. Information about the project size[5] can be found in Table 3.1.

BUSYBOX Busybox combines different unix-software, that is considered to be standard, into a single program. Its main use-case is to run on small systems, like embedded systems or inside docker containers to give those systems unix functionality, without having to install a complete unix environment. It was first created by Bruce Perens in 1996. After it was made public in 1999, the project has gone through the hands of multiple maintainers until 2006, when Denys Vlasenko took over maintanence. The project is considered the smallest of the three projects covered in this thesis, because its total number of authors is the smallest[5].

OPENSSL OpenSSL is an open-source software implementing encryption algorithms as well as generation and management functionality for digital certificates. The project was built as a fork of SSLeay, an early open-source SSL implementation, in 1998. In 2019, 17 developers in total have commit authorization, other contributors have to submit their code contributions via pull requests. Like most older software projects, OpenSSL relied on mailing lists until it

1 https://www.busybox.net/
2 https://www.openssl.org/
3 https://www.qemu.org/
4 With an exception of OpenSSL
5 For the data that was used for this thesis. The projects have undergone further development in the meantime.

started to transition to the Github issue system for communication and organization in 2013. With slightly more than 8000 authors, for mailing-lists and code contributions combined, OpenSSL is the second project in terms of author-based project size in this thesis[19].

QEMU The last project in the list, QEMU, was started in 2003 by Fabrice Bellard. The name QEMU is an acronym and means "Quick Emulator". QEMU is used for hardware virtualization and even was partially used by VirtualBox. Like Busybox, QEMU still uses mailing-lists as a communication medium. Regarding the number of commits and mails, this project is by far the biggest project in the list[20].

Table 3.1: Overview of all projects used in this thesis

Name	Initial release	Number of commits	Number of mails	Number of authors
Busybox	1999	27285	46158	3091
OpenSSL	1998	42753	37723	8079
QEMU	2003	123380	740429	9150

3.2 DATA PREPARATION

In order to answer the previously posed research questions, data is needed first, namely mailing list data as well as code-commit data. Other data that can be used to model author networks in an open-source software projects could be issue-data, but since all analysed projects use or were using mailing-lists as their primary medium for communication, only a few years of issue-data would be available only for OpenSSL. Networks for code-commit data in this thesis only use the file-based relation, meaning that authors who edited the same file, connect with each other. Other types of author relation for commit-data based networks are function-based relation, i.e. only authors who edit the same function inside a file connect with each other, and the feature-based relation, i.e. authors who commit code for the same feature connect to each other. There are multiple reasons, the file-based relation was chosen. First, using more than one commit-based author relation could easily blast the length of this bachelors thesis and second, before analysing a more fine-grained author relation like the function-based or feature-based relation, a more general relation like the file-based relation should be analysed in order to get an overview about the networks.

As mentioned previously in Chapter 1, this data is collected using the codeface framework [6]. An example for mail data can be found at Listing 3.1 and an example for commit data can be found at Listing 3.2.

Listing 3.1: An example for mail data. Column names for the semicolon seperated data from left to right: Name of the author; Mail address of the author; Message ID; Date of dispatch; Timezone offset; Mail subject; Thread ID

```
"Author name"; "author.mail@example.net"; "<434571BC.8070702@example.net>";
"2003-10-14 16:06:13"; "200"; "Re: [PATCH] syslog,logread,etc."; "73#1000"
```

6 See: https://github.com/siemens/codeface

Listing 3.2: An example for commit data. Column names for the semicolon seperated data from left to right: Commit ID; Author submition date; Author name; Author mail; Commit date; Committer name; Committer mail; Commit Hash; Number of changed files; Lines added; Lines removed; Difference lines added and lines removed; Name of changed file; Name of function; Type of change (in this case "Function"); Lines changed in this function

```
"4572950"; "1999-10-05 16:24:54"; "Author name"; "author.mail@example.net";
"1999-10-05 16:24:54"; "Committer name"; "commiter.mail@example.net";
"cc8ed39b240180b58810784f844e253263594ac3"; "113"; "24504"; "0"; "24504";
"archival/gzip.c"; "bi_windup"; "Function"; "13"
```

This data is used to build networks using the coronet library[7] with its nodes being the authors. Edges have a different meaning depending on the type of data. In case of mail data, edges are formed between authors, who participated on the same mailing list thread. So when, e.g. author John Doe writes a mail on a thread ten other people already participated on, an edge between him and each of these other ten authors will be created. For commit data, a file-based approach is chosen. This way, when an author edits a file, edges between this author and every other author who previously edited this particular file are created. So e.g. author John Doe adds a line to the file main.cpp, which was already edited by ten other people before, an edge between John's node and every of the other ten nodes is then created. Time and date of each action are preserved, so the network can be split in the next step.

As already mentioned in Chapter 2, splitting the network is important for analysing the time evolution of said network. If the network would not be split, we could only see the whole state of the network at the time of data acquisition, because all edges between developers are already created. Such a network does not fit well for the type of analysis that is conducted in this thesis, because the analysis requires information about the networks evolution, i.e. we need to know which authors did connect with each other during which time periods, so transitions between those time periods can be analysed. Networks are split using multiple configurations, i.e. the timeframe length as well as the splitting method is adjusted. The chosen values for timeframe lengths are three months, six months and nine months, which has multiple reasons, the first one being that Barabási suggested to use short timeframe lengths, using lengths of one year himself[13] and the other being that all three timeframe lengths are dividable by three, meaning that both, six-month and nine-month timeframe lengths can be compared with the three-month timeframe length by simply mapping one timeframe of the six-month timeframe length onto two timeframes of the three-month timeframe length covering the exact same time period or mapping one timeframe of the nine-month timeframe length onto three timeframes of the three-month timeframe lenght covering the exact same time period. Additionally, each length is also split using sliding windows, i.e. for each two adjacent timeframes, a third one starting in the middle of the first timeframe and ending in the middle of the second timeframe is created; this way, information loss by splitting in the midst of some interesting activity can be mitigated. In total, there are six timeframe length configurations (x months as well as x months with sliding windows). Node-based and edge-based splitting[8] is combined with each of the timeframe lengths, resulting in 12 final splitting configurations. For each software project, this means that both mail and commit

7 See: https://github.com/se-sic/coronet
8 See paragraph 2.2

based networks are split by those 12 splitting configurations, totaling to 24 split network types. With these networks, the following research questions can be addressed.

3.3 RQ1: HOW DO NEW AUTHORS CONNECT TO A DEVELOPER NETWORK?

For the connectivity analysis, i.e. finding out whether newcomers first connect to other newcomers or more active developers, we first have to define classifications for nodes and edges. For nodes, there are only two types, namely *new* and *old*. A *new* node is a node that did not appear in the network in any earlier timeframe, while an *old* node has been present in any earlier timeframe. This classification has to be done in order to find out, which developer in the network is a newcomer and which developer is not. This is done by creating a *new* node for a newcomer and an *old* node for a developer who also has been active in the last timeframe. Edges are classified as the types *a*) old-old, *b*) new-old and *c*) new-new. *Old-old* defines an edge between two *old* nodes, a *new-old* edge is an edge between a *new* node and an *old* node and the last type, *new-new*, defines an edge that is created between two *new* nodes.

In order to not only look into each timeframe seperately but at the whole history of the software-project instead and to not further increase the complexity of the analysis we will be using edge-based splitting. In an already built and split network, we first define an empty set $A = \emptyset$ and iterate over all timeframes $t \in T$ of this network. For each timeframe t, we define three counters, one for each edge type which are defined as above. With E_t, the set of all edges in the timeframe t, we then look at each edge $e = \langle v_1, v_2 \rangle \in E_t$. Now we can have three cases:

$$v_1 \in A \land v_2 \in A \,, \tag{3.1}$$

$$(v_1 \in A \land v_2 \notin A) \lor (v_1 \notin A \land v_2 \in A) \,, \tag{3.2}$$

$$v_1 \notin A \land v_2 \notin A \tag{3.3}$$

If e matches equation 3.1, we increase the *old-old* counter. If e matches equation 3.2, we increase the *new-old* counter. If none of the cases before match, e must match equation 3.3 and therefore we increase the *new-new* counter. With V_t, the set of all nodes in the timeframe t, we set $A = A \cup V_t$ at the end of each iteration. Now that we have each type of edge for each timeframe, we can calculate the proportion of each edge type in each timeframe as well as the proportion of each edge type in the whole project network. Using these proportions, we can find out whether newcomers prefer to connect to other newcomers or to developers who are already active in the projects community.

3.4 RQ2: DO THESE PROJECTS EVOLVE BASED ON PREFERENTIAL ATTACHMENT?

In order to find out whether a project evolves based on preferential attachment, we again have to split the network into timeframes. As a rough overview, in order to find out if the network evolved based on preferential attachment between two timeframes, the distribution of new edges per vertex degree must be checked whether or not it can be fit on a power-law

distribution[13]. For such a fitting, we have to find out the vertex degrees of authors from the first timeframe and assign weights onto them based on the amount of edges that are acquired

Like in the connectivity analysis in the previous section, we iterate over the edge-based split timeframes T in ascending order, only that we start at the second timeframe t_2, because for each timeframe t_n we need t_{n-1} in order to find out which developers are newcomers and which developers were already active in t_{n-1}. If we would start at t_1, all authors in t_1 would be considered newcomers, and no information about the previous vertex degrees would be available, resulting in all new edges being accounted towards the vertex degree zero. We can get the set of authors, that have not been in t_{n-1} with $V_{\text{new}} = V(t_n) \backslash V(t_{n-1})$ and the set of authors, that was also in t_{n-1} with $V_{\text{old}} = V(t_{n-1}) \backslash V(t_n)$ so $V_{\text{old}} \cap V_{\text{new}} = \emptyset$. Edges from nodes within V_{new} to nodes within V_{old} are called *external* and edges from nodes within V_{old} to nodes within V_{old} are called *internal* by Barabási[13]. Edges from V_{new} to V_{new} will be ignored, because there is no vertex degree in the previous timeframe and therefore the new edge could not be accounted to any vertex degree. Next we have to count edges per vertex degree, i.e. for each author in V_{old} with a vertex degree k, we increase the counter c_k by one. With this counter c_k, we can find out how much new connections an author with a previous vertex degree k acquired and therefore find out whether or not those authors with a high vertex degree also acquire most new edges. All counters c_k combined yield the degree distribution $\Pi(k)$, that is only dependent on k[13].

Since the degree distribution $\Pi(k)$ must follow a power-law, we must fit a power-law distribution to $\Pi(k)$[13]. For fitting a power-law distribution to our data, we must find out a value for the lower bound x_{min} first, because typically, the lower end of a distribution contains "non-power-law behaviour"[8], so by estimating x_{min}, we estimate where power-law behavor starts[8]. In order to estimate x_{min}, we choose a value $\hat{x}_{\text{min}}$ that creates the highest similarity between $\Pi(k)$ and a power-law distribution. This similarity can be measured using the Kolmogorov-Smirnov statistic which is used for finding out the maximum distance between two CDFs[8]. With $x_{\text{min}} = \hat{x}_{\text{min}}$ for $\hat{x}_{\text{min}}$ that creates the highest similarity, we estimate the scaling parameter α using a maximum likelihood estimation, which then is used to fit a power-law distribution with this scaling parameter α to $\Pi(k)$. In order to find out whether or not the power-law distribution with scaling parameter α fits $\Pi(k)$, we use the Kolmogorov-Smirnov test for finding out the maximum distance between $\Pi(k)$ and the power-law distribution. This test returns a p-value, that we call KS.p[16]. When KS.p is less than 0.05, we say that the Kolmogorov-Smirnov test did not confirm a power-law fit for $\Pi(k)$[8].

The classification whether or not the timeframe t_n follows the preferential attachment model, is done via the following function:

$$\text{pref}(t) = \begin{cases} 1, & \text{if KS.p } \geq 0.05 \\ 0, & \text{otherwise} \end{cases}$$

After applying this function on each timeframe $t_n \in T$, we know during which timeframes in the evolution of the project, the project growth followed the preferential attachment model. With this information, we can apply further analysis to find out relationships between different parameters and metrics.

3.5 RQ3: IS THERE A RELATION BETWEEN PREFERENTIAL ATTACHMENT AND THE AMOUNT OF ACTIVITY IN THE PROJECT?

To find out about relationships between preferential attachment and project activity, we have to retrieve the activity for each timeframe, i.e. counting all mails or commits in the time period of each edge-based split timeframe $t_n \in T$. For this, we introduce a variable a_n, being the number of mails or commits[9] for the n-th timeframe. Using $a_{\min}$ as the smallest a_n, i.e. the smallest number of mails or commits in all timeframes, and $a_{\max}$ as the highest a_n, i.e. the highest number of mails or commits in all timeframes, in the given split network, we can calculate the normalized activity per timeframe using the following equation:

$$\text{Norm}(a_n) = \frac{a_{\max} - a_{\min}}{a_n - a_{\min}} \tag{3.4}$$

For the n-th timeframe, $\text{Norm}(a_n)$ is n-th timeframes percentized activity in relation to the maximum activity in the whole network. This normalization ensures, that we can look at the activity for each timeframe relative to the total activity in the project, making it easier to interpret the resulting data. We then can compare for each timeframe t_n the preferential attachment data (that we computed in the last section) as well as its corresponding $\text{Norm}(a_n)$ and try to find a correlation between those two metrics, which can be useful in further understanding when and why a open-source software projects network evolved based on preferential attachment. There are three possible outcomes, being a) for each timeframe t_n, that follows the preferential attachment model $\text{Norm}(a_n)$ is remarkably high and for each timeframe t_n that does not follow the preferential attachment model $\text{Norm}(a_n)$ is remarkably low, b) for each timeframe t_n, that follows the preferential attachment model $\text{Norm}(a_n)$ is remarkably low and for each timeframe t_n that does not follow the preferential attachment model $\text{Norm}(a_n)$ is remarkably high or c) no correlation can be observed using this method.

In case of item a), the conclusion would be, that whenever the project's activity increases, its networks evolution follows the preferential attachment model. In case of item b), the complete opposite would be the case; whenever the project's activity decreases, its networks evolution follows the preferential attachment model. This case, purely intuitive speaking, would be quiet paradox, because when there is not much activity, there also are not that much potentially new edges that an already highly connected node could get linked by. In case of item c), no statement about a relation between preferential attachment and project activity can be made.

3.6 RQ4: WHAT IMPACT DO DIFFERENT TIMEFRAME LENGTHS HAVE ON ANALYSIS RESULTS?

To find out differences and similarities between different time-window lengths, we compare the results of the preferential attachment analysis for different timeframe lengths with each other. The timeframes that are compared will all be edge-based split. Since all timeframe lengths are dividable by three, we can group timeframes of lower timeframe lengths together and compare them to a timeframe with a higher timeframe length, e.g. we can group the first two three-month timeframes with the first six-month timeframe. When comparing different timeframe lengths, one has to look out for similarities or differences, e.g. when a group of

9 Depending on the network configuration

two three-month timeframes has one timeframe that follows the preferential attachment model and one timeframe that does not follow the preferential attachment model, it might be interesting to see whether or not the six-month timeframe during the same time does follow the preferential attachment model. Another setting that would require further investigation, would be, e.g. when a group of three-month timeframes all evolved based on preferential attachment, but the associated six-month or nine-month timeframes did not evolve based on preferential attachment.

Since only edge-based splitting was used for the previous analyses, at least an overview about differences between results from edge-based splitting and node-based splitting can be given. As a quick refresher, since node-based splitting, in theory, returns results used for microscopic analysis on each timeframe, all analyses regarding preferential in this thesis use edge-based splitting in order to get results based on a more macroscopic layer. In order to find out similarities between both splitting types, the classification from *RQ2* can be used for each network configuration, both edge-based and node-based split, so for each configuration, e.g. three-month, file-based, non-sliding-window, two results can be retrieved, one for the previously processed edge-based network and one for the associated node-based network. An easy approach for finding similarities in results of differently split networks is comparing the preferential attachment results of each timeframe $t_n \in T_E$ of the edge-based split network with the preferential attachment results of the timeframe $t_m \in T_N$ of the node-based split network with $n = m$. Since these two networks are created on the same data with the same timeframe length and sliding-window parameter, they also share the same amount of timeframes covering the same periods of time, therefore a one-to-one comparison between whether or not a timeframe evolves based on preferential attachment can be conducted. We introduce a variable c being the amount of timeframes $t_n \in T_E$ matching their preferential attachment property with the timeframes $t_m \in T_N$ covering the same periods of time, i.e. c is the amount of timeframes fulfilling $\text{pref}(t_n) = \text{pref}(t_m)$. We then get the fraction p_c of matches by calculating

$$p_c = \frac{c}{|T|}. \tag{3.5}$$

The closer p_c is to one, the greater is the similarity between those two split types are in the chosen configuration.

4 EVALUATION

In this chapter, three different projects are analysed using the methods that are described in Chapter 3. We will go over computed graphs and data and discuss the general growth model of those projects, namely Busybox, OpenSSL and QEMU. These projects were chosen because they differ from each other in project size and in age.

4.1 RESULTS

In this section, visualizations and tables will be shown and their legends and their axes will be described. Those legend and axis descriptions are analogous to each project, so it will only be explained for the first project. This section only shows the resulting visualizations and tables, discussion will be found in Section 4.2.

4.1.1 *RQ1: How do new authors connect to a developer network?*

The graphs shown here describe new edges per nine-month timeframe, categorized by their type. There are two types of graphs, one showing the absolute amount of new edges in the given timeframe and the other showing the relative amount of each edge type in each timeframe. As for the edge type, it will be distinguished between external edges, internal edges, as well as new-to-new edges[1]. On the x-axis, we can see the start date of each timeframe, where each bar describes a whole timeframe with the length of nine months. The y-axis describes the amount of edges. Each bar on the y-axis is seperated into a maximum of three colors: red for external edges, green for internal edges and blue for new-to-new edges. In the absolute plot, the total bar height determines the total amount of edges acquired during the according timeframe. Each subdivision in a bar describes the proportion of the edge-type, that is encoded by the color of the sub-bar.
The bars in the relative plot always have a total of 1, meaning that each relative bar represents 100% of the total links of the according timeframe where, as in the absolute visualization, each subbar describes the percentiled proportion of each edge-type in the given timeframe. Since there is no information about the amount of edges in this type of visualization, it cannot be used for comparison of different timeframes in terms of absolute edges. It is only useful for comparing the edge types inside a singular timeframe and the proportions of each edge type between two different timeframes.

The networks were split in a non-sliding-window approach with the edge-based splitting type. Visualizations shown here are grouped into two-by-two squares, so it does not disrupt the reading flow. Everything except the dates on the x-axis are still readable this way. If you are interested in the fully sized visualizations, you can take a look at Section A.1. The top row shows the cochange (commit data) visualizations, the bottom row shows the mailing

1 See Section 3.3

list visualizations. The left figure in each row shows the absolute data for each timeframe, the right figure shows the relative data for each timeframe. The visualizations for Busybox are shown in Figure 4.1, the visualizations for OpenSSL are shown in Figure 4.2 and the visualizations for QEMU are shown in Figure 4.3.

BUSYBOX The file-based network for Busybox mostly consist of internal edges. While there are new authors, who connect with internal authors, they only create a fraction of all new edges (~40% at most). For new-to-new edges, this edge type first starts to come up at the end of 2011 with at most ~4% of all new edges in 2013. With just some exceptions, with a huge amount of new edges, the total amount of new edges in the file-based got steadily more.

The mail-based networks for Busybox look are a bit different: the proportion of internal edges is at maximum ~45%, with a minimum of ~9% in 2003. New-to-new edges make up for more edges in all networks when compared to the file-based networks for Busybox. In 2003, new-to-new edges constitutes around 32%, which rapidly got less in the next years. After 2007, new-to-new edges made up for around 8% constantly with only a few fluctuations. Proportionally, external edges are responsible for most new edges in the mail-based network. With a proportion of more than 50% of all new edges, external authors seem to mostly connect to internal authors. In contrast to the file-based network for Busybox, the total amount of new edges constantly decreased after the end of 2005.

OPENSSL The file-based network of OpenSSL has a similar size as the file-based Busybox network in terms of total amount of new edges from 2002 until 2012. The last timeframe in the graphs in Figure 4.2 should be ignored due to its incompleteness, that is caused by the timeframe ending before it reached the nine-month length. The reason for this, is that data acquisition had to be done at some time, and in case of data acquisition for OpenSSL, the data was collected just a short time into the last nine-month timeframe.

New-to-new edges are only present in the end of 2013 with less than 2% of the total new edges in that timeframe. The proportion of external edges in contrast to the total amount of new edges goes from less then 2% up to ~30%. The biggest part of new edges in the file-based network are internal edges with a minimum of ~60% of all new edges.

In the mail-based network, the last four timeframes are empty due to the transition from mailing-list based communication to communication and organization via Github issues. The rest of the network looks similar to the mail-based network of Busybox, having new-to-new edges with a proportion of ~4% to ~30% of all new edges in every timeframe.

QEMU For QEMU, the first two timeframes in the file-based network only consist of external edges, with a very low amount of total new edges. The network starts to really grow in 2009, when total new edges jump from about 2000 to over 5000. New-to-new edges are not present in any file-based network timeframe and external edges make up for ~1% to ~10% of all new edges.

Like in the networks of Busybox and OpenSSL, the percentage of external edges as well as the percentage of new-to-new edges is much higher in the mail-based network. Compared to the file-based network, the amount of new edges in 2003 is already relatively high, having close to 2000 new edges. The percentage of new-to-new edges in this timeframe equals to about 73%, while the percentage of external edges during this timeframe is about 25% and internal

edges making up for only ~2%. This behaviour changes rapidly after this timeframe, when external edges make up for the majority of all new edges, being ~55% to ~62%. During this time, the percentage of new-to-new edges only ranges from ~20% to ~5%, falling constantly.

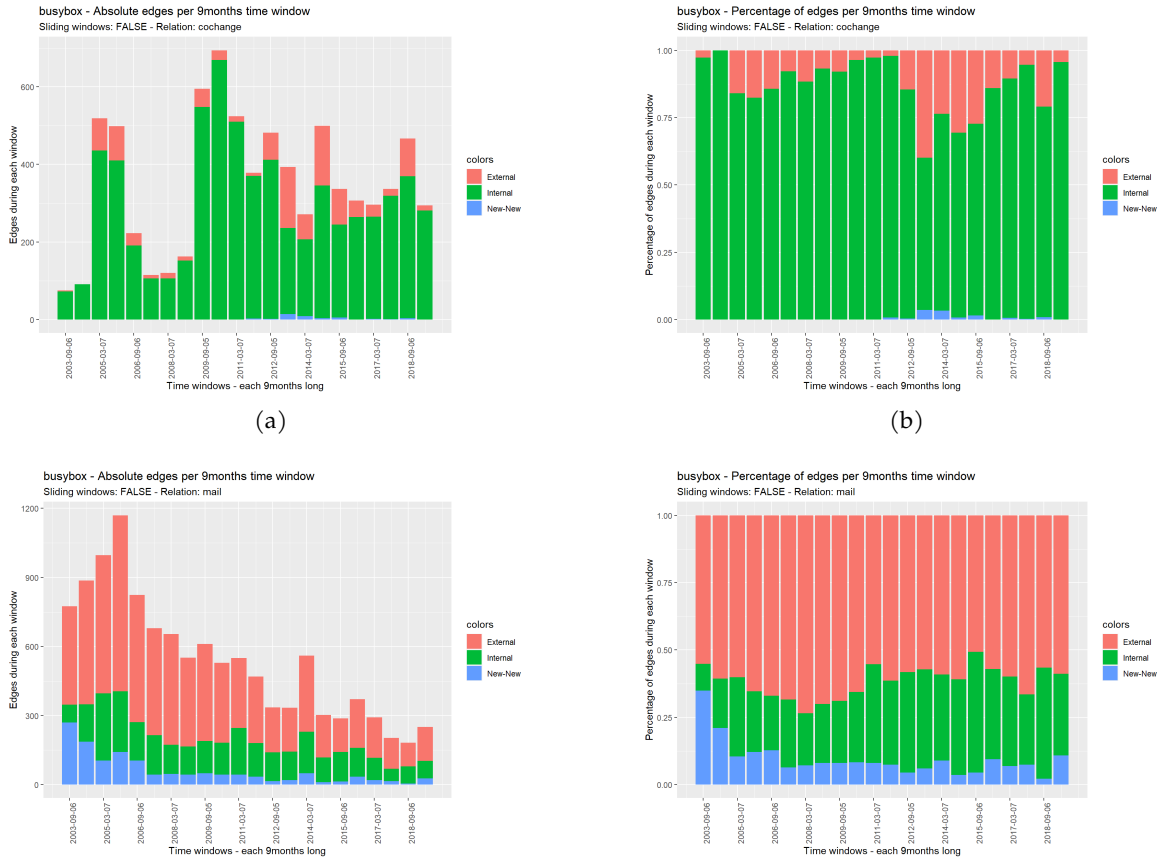


Figure 4.1: The creation rate for new edges for both commit and mailing list data for the Busybox project: In (a), the absolute file-based edges for the Busybox project can be seen. (b), shows the relative amount of file-based edges for each edge type for the given timeframe. In (c), the absolute mail-based edges for the Busybox project can be seen. (d), shows the relative amount of mail-based edges for each edge type for the given timeframe.

4.1.2 *RQ2: Do these projects evolve based on preferential attachment?*

There are two types of visualization, that can be used as a result for this question: the cumulative function summing all edges for a given vertex degree k on a logarithmic scale[13] and the plots showing whether or not the degree distribution in a timeframe was fit to a power-law. For the cumulative function, timeframes for three different points in time of the acquired data are shown: one for the beginning of the acquired data, one for the center of the acquired data and one for the end of the acquired data. Each figure for the cumulative function shows rows of plots, which correspond to consecutive timeframes. The three visualizations for each timeframe show (from left to right) *a*) the cumulative function for the total links of the given

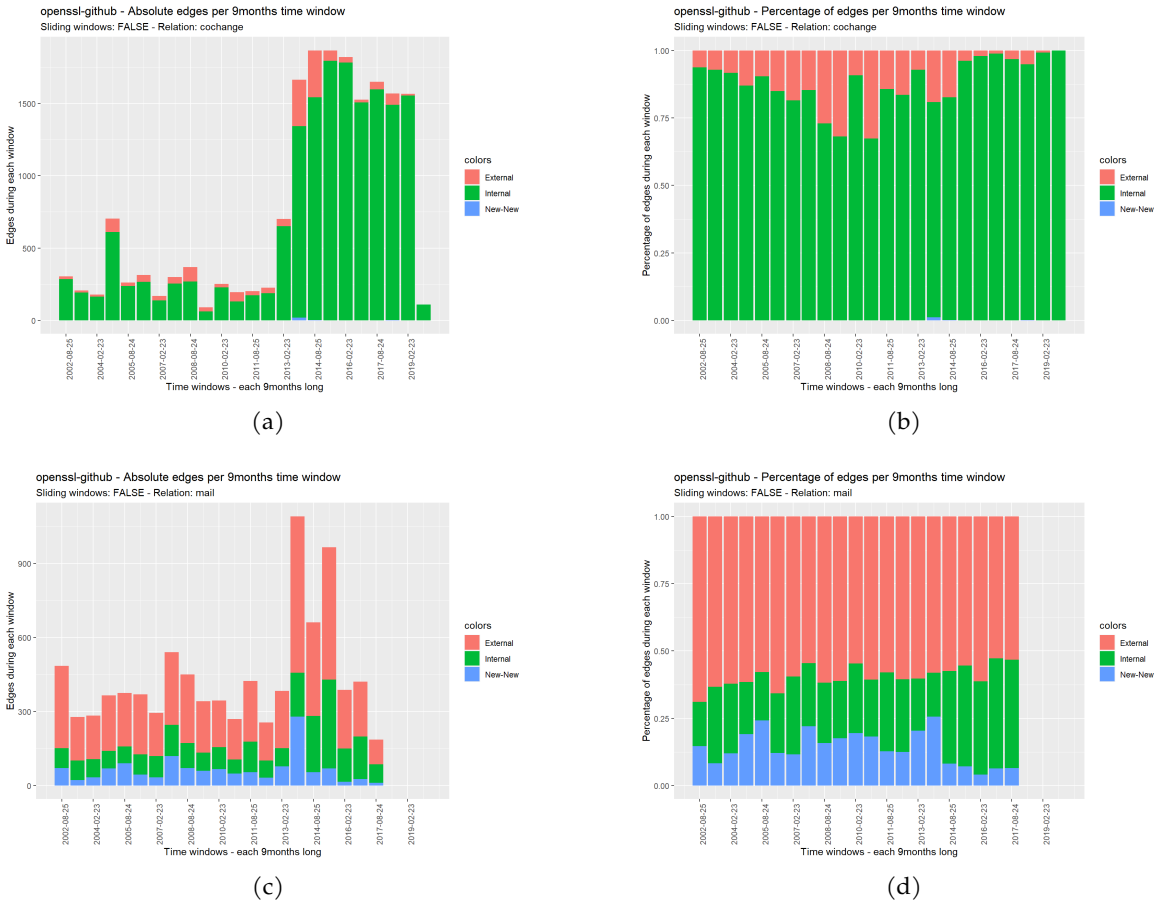


Figure 4.2: The creation rate for new edges for both commit and mailing list data for the OpenSSL project: In (a), the absolute file-based edges for the OpenSSL project can be seen. (b), shows the relative amount of file-based edges for each edge type for the given timeframe. In (c), the absolute mail-based edges for the OpenSSL project can be seen. (d), shows the relative amount of mail-based edges for each edge type for the given timeframe.

timeframe, i.e. both internal and external links combined, b) the cumulative function for the internal links of the given timeframe and c) the cumulative function for the external links of the given timeframe. All three plots read the same, they only differ in the type of edge they are visualizing. Again, both commit data and mailing list data is visualized, but both visualizations are read the same. The x-axis of each plot describes the vertex degree of each corresponding dot. Each dot is not a singular node in the network, but the sum of all nodes with the given vertex degree. The y-axis describes the accumulated amount of the proportion of the acquired links by the given vertex degree for all acquired links in the given timeframe. Since the data is accumulated, a group of nodes with the vertex degree k_t having a rate of acquiring new links of y does not necesarrily mean that this group of nodes acquired $y * 100\%$ of all new links in this timeframe, it only means that the rate of acquiring new links for the vertex degree k_t is the sum of the rate of acquiring new links for the vertex degree k_{t-1} and the actual proportion of the new links in the given timeframe for the vertex degree k, e.g. the rate of acquiring new links for the vertex degree k_{t-1} is 0.4 and the proportion of new links

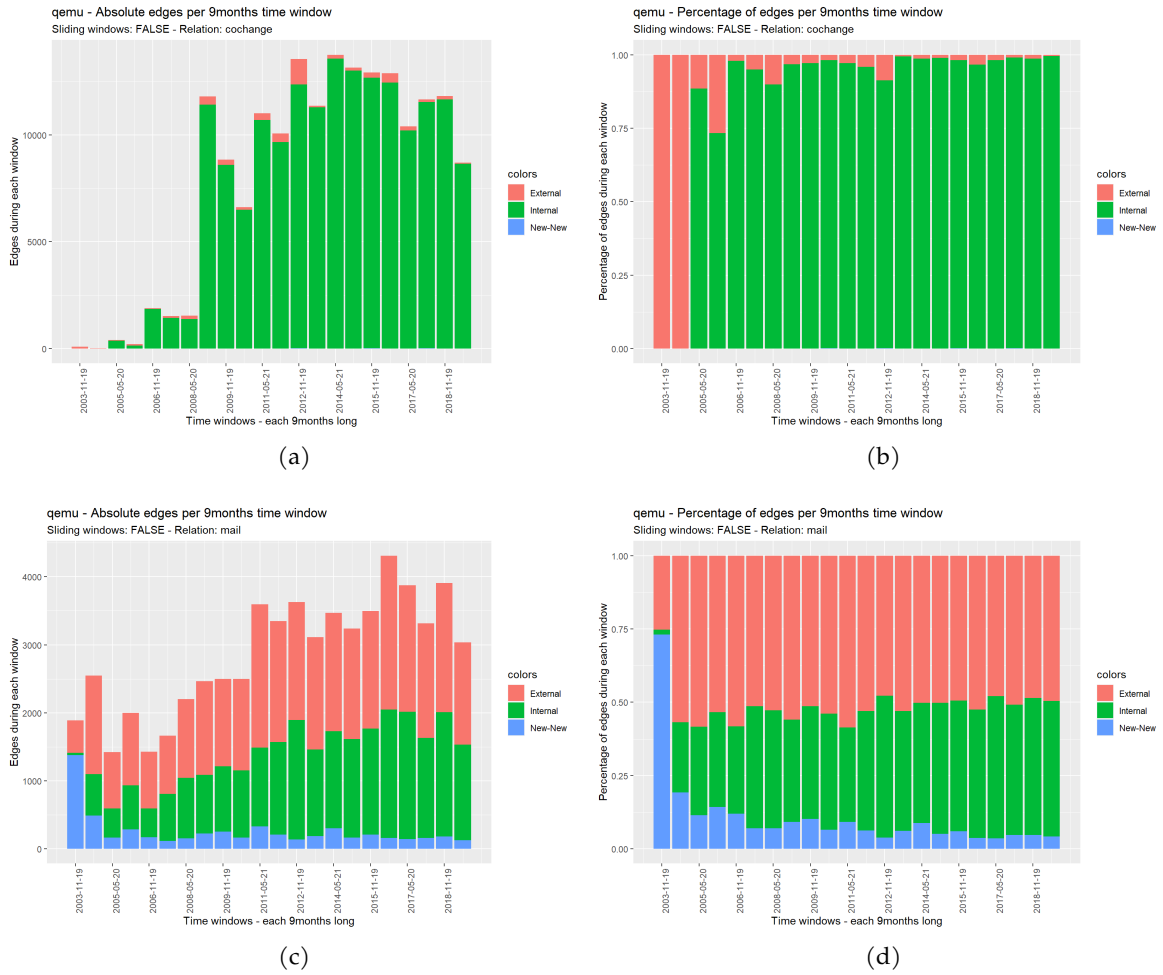
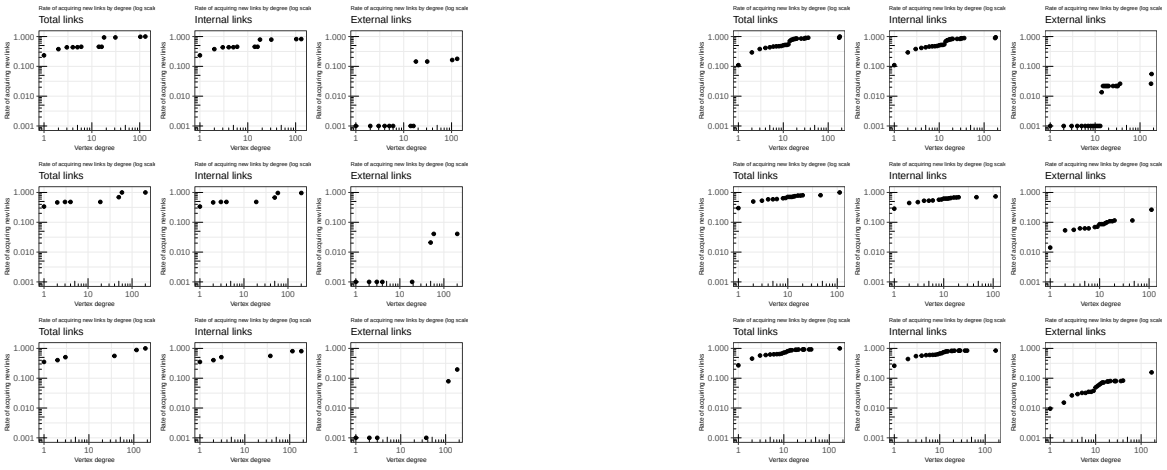


Figure 4.3: The creation rate for new edges for both commit and mailing list data for the QEMU project: In (a), the absolute file-based edges for the QEMU project can be seen. (b), shows the relative amount of file-based edges for each edge type for the given timeframe. In (c), the absolute mail-based edges for the QEMU project can be seen. (d), shows the relative amount of mail-based edges for each edge type for the given timeframe.

in the current timeframe for the vertex degree k is 0.1, the resulting rate of acquiring new links for k is 0.5. This behaviour allows us, to look at the shape of the resulting plot and its slope. A small slope, where small vertex degrees already would have a high rate would be the result of a network, where nodes with a small vertex degree acquire most of the new links, whereas a steep slope, where the rate for small vertex degrees is significantly smaller than the the rate for large vertex degrees would be the result of a network where a highly connected node also has a high rate of acquiring a new link; the latter example would be an indicator for the network evolving based on the preferential attachment model. An example for this type of visualization can be seen in Figure 4.4 for the file-based network of the Busybox project. In this example, we can see visualizations for the rate of acquiring new links for the three different edge-types: total edges, internal edges and external edges. The union of internal edges and external edges results in the set of total edges and the sum of rate of acquiring new

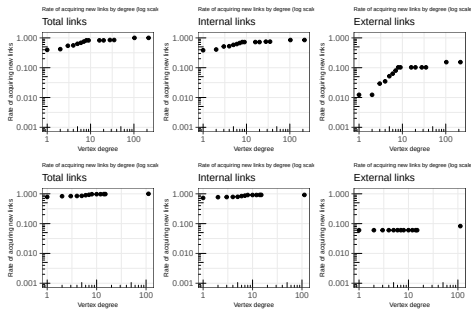
links (the y-axis) from internal links and external links results in the rate of acquiring new links for the total edges. In the first row of Figure 4.4b, we can see for the total links, that nodes having a vertex degree of one, already have a rate of acquiring new links of greater than 0.1. For nodes with a vertex degree of ten, the rate of acquiring new links is more than 0.5 and nodes with a vertex degree of more then 100, the rate of acquiring new links is 1. Nodes with the highest vertex degree always have a rate of acquiring new links of 1, because the values are cumulative. In this example, the rate of acquiring new links for the highest vertex degree is 1, even though nodes with a vertex degree of about 50 already have a rate of acquiring new links of about 0.9. When we try to make assumptions about whether or not this timeframe for the total-edge network evolves based on preferential attachment, we would see that nodes with a low vertex degree already have a comparably large rate of acquiring new links and that if we would fit a straight line onto the values of the plot, the slope would not be steep, therefore the timeframe should not be evolved based on preferential attachment. Other information that can be gathered from the first row of Figure 4.4b, is that the rate of acquiring new links for internal links and external links is different. For the external-edge timeframe, there seems to be no link acquisition for nodes with a vertex degree of less then about 15. For nodes between vertex degree 15 and 50, only a few vertex degrees have acquired new links, because there is no slope in this area for most vertex degrees. For the highest vertex degree in the external-edge timeframe, the rate of acquiring new links is not 1, because it is normalized over the total links and not only the external links.

All of the visualizations for this type of plot can be found in the appendix beginning at Figure A.13.



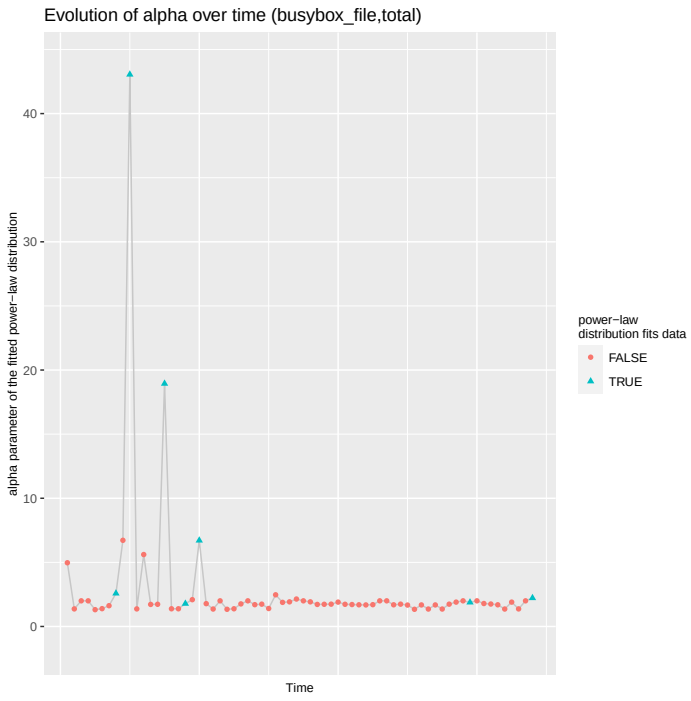
(a)

(b)

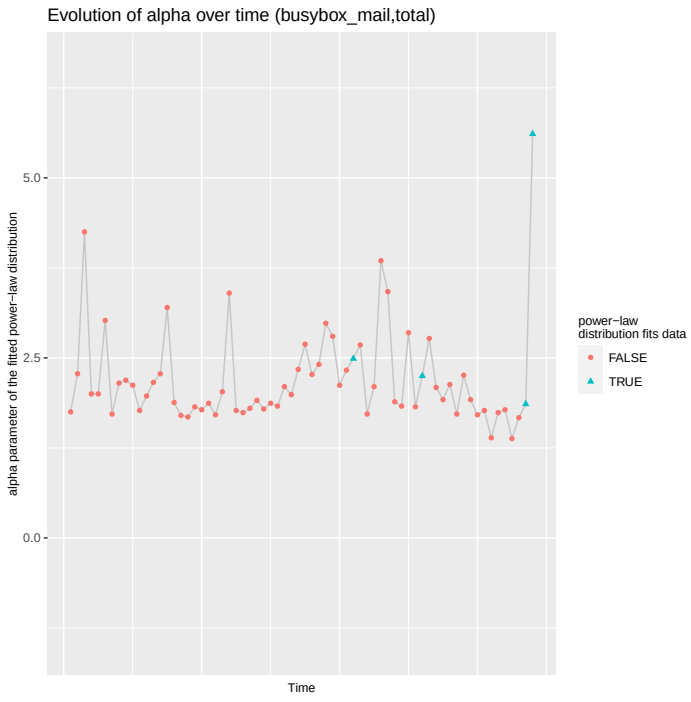


(c)

Figure 4.4: The cumulative function for the three-month timeframe length of the commit-based data (edge-based split) for Busybox: In (a), visualizations for three three-month timeframes from 2003-03-08 until 2003-12-07 is shown. In (b), visualizations for three three-month timeframes from 2011-06-07 until 2012-03-07 is shown. In (c), visualizations for three three-month timeframes from 2019-09-06 end of record is shown.



(a)



(b)

Figure 4.5: The comparison between powerlaw-fit exponent α and preferential attachment for Busybox: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

The other type of plot shows multiple information at once, namely whether or not the degree distribution of the preferential attachment data of a timeframe can be fit on a power-law distribution and the exponent of the fitted power-law distribution. The x-axis describes the timeline of the project, mapping each point (or triangle) in the graph onto a timeframe. The y-axis describes the α-value for the fitted power-law distribution, i.e. the exponent of the fitted distribution. A timeframe can have one of three different icons, *a*) a red circle, denoting that the timeframe did not fit a power-law distribution, thus did not evolve based on preferential attachment, *b*) a blue triangle, denoting that the timeframe did fit a power-law distribution, thus did evolve based on preferential attachment and *c*) nothing on the line to denote that there was not enough data to fit it to a distribution. An example for this visualization can be found in Figure 4.5. All visualizations of this type combined can be found in the appendix starting at Figure A.19. The graphs, both commit-based and mail-based for the split networks with a length of three months can be found for Busybox in Figure 4.5, for OpenSSL in Figure A.20 and for QEMU in Figure A.21.

BUSYBOX For the total new edges of the file-based network of Busybox, the minimum percentage of power-law fits is 8% for the three-month timeframe length and the maximum percentage of power-law fits is 13% for the nine-month timeframe length. The minimum percentage of power-law fits for only internal edges is 38% for the six-month timeframe length and the maximum percentage is 44% for the three-month timeframe length. For the external edges, the minimum percentage of power-law fits is 2% for the six-month timeframe length and the maximum percentage of power-law fits is 9% for the nine-month timeframe length.

In the mail-based network, the minimum percentage of power-law fits for all new edges is 0% for the nine-month timeframe length and the maximum percentage is 5% for the three-month and the six-month timeframe length. For the internal edges, the minimum percentage of power-law fits is 59% for the nine-month timeframe length and the maximum percentage is 82% for the three-month timeframe length. As for the external edges, the minimum percentage of power-law fits is 0% for the nine-month timeframe length and the maximum percentage is 5% for the six-month timeframe length.

These values can be found in Table 4.1.

Table 4.1: Busybox: Proportion of power-law fits

		Edge type		
		Total	Internal	External
File	Minimum	8%	38%	2%
	Maximum	13%	44%	9%
Mail	Minimum	0%	59%	0%
	Maximum	5%	82%	5%

OPENSSL In the file-based network, the minimum percentage of power-law fits for all new edges is 0% for the six-month timeframe length and the maximum percentage is 8% for the nine-month and the six-month timeframe length. For the internal edges, the minimum percentage of power-law fits is 5% for the six-month timeframe length and the maximum

percentage of power-law fits is 20% for the three-month timeframe length as well as for the nine-month timeframe length. The minimum percentage of power-law fits for only external edges is 11% for the three-month timeframe length and the maximum percentage is 19% for the six-month timeframe length.

For the total new edges of the mail-based network of OpenSSL, the minimum percentage of power-law fits is 4% for the nine-month timeframe length and the maximum percentage of power-law fits is 15% for the three-month timeframe length. As for the internal edges, the minimum percentage of power-law fits is 66% for the nine-month timeframe length and the maximum percentage is 88% for the three-month timeframe length. For the external edges, the minimum percentage of power-law fits is 4% for the nine-month timeframe length and the maximum percentage is 33% for the three-month timeframe length.

Table 4.2: OpenSSL: Proportion of power-law fits

		Edge type		
		Total	Internal	External
File	Minimum	0%	5%	11%
	Maximum	8%	20%	19%
Mail	Minimum	4%	66%	4%
	Maximum	15%	88%	33%

QEMU For the total new edges of the file-based network of QEMU, the minimum percentage of power-law fits is 1% for the three-month timeframe length and the maximum percentage of power-law fits is 9% for the nine-month timeframe length. The minimum percentage of power-law fits for only internal edges is 2% for the three-month and the six-month timeframe length and the maximum percentage is 4% for the nine-month timeframe length. For the external edges, the minimum percentage of power-law fits is 22% for the nine-month timeframe length and the maximum percentage of power-law fits is 35% for the six-month timeframe length.

Regarding the mail-based network, the percentage of the power-law fits of all edge types combined for all three timeframe lengths is 0%. The percentages of power-law fits for the mail-based network are 0% as the minimal percentage for the nine-month timeframe length and 6% for the three-month timeframe length. Looking at the external edges, there are no power-law fits for the degree distributions of the split network.

Table 4.3: QEMU: Proportion of power-law fits

		Edge type		
		Total	Internal	External
File	Minimum	1%	2%	22%
	Maximum	9%	4%	35%
Mail	Minimum	0%	0%	0%
	Maximum	0%	6%	0%

4.1.3 *RQ3: Is there a relation between preferential attachment and the amount of activity in the project?*

The relation between preferential attachment and the activity of a project, i.e. the number of commits and mails per timeframe, can be shown as an evolutionary graph. On the x-axis, the time-evolution describes and on the y-axis, the proportional amount of activity, i.e. the proportion of the amount of commits or mails in the given timeframe in relation to the maximum and minimum of commits or mails in the whole project. This means, that the timeframe with $y = 0$ represents the timeframe with the minimum of acitivity for the given analysis, while the timeframe with $y = 1$ represents the timeframe with the maximum of author activity for the given analysis. Additionally, each point on the graph is labeled either with a red circle or a blue triangle. This labeling, same as for *RQ2*, describes wheter or not the timeframe evolved based on the preferential attachment model. The visualizations shown here are all for the networks that were split with lengths of three months. This way, a bigger picture of the whole evolution can be seen. To get an overview of the projects evolution, visualizations for Busybox can be found in Figure A.22, for OpenSSL in Figure A.23 and for QEMU in Figure A.24. The results for this research question are computed on the total edges, i.e. internal and external edges combined, so as not to increase the complexity of the thesis dramatically.

BUSYBOX For the file-based network of Busybox, there was no configuration in which a correlation between activity and preferential attachment, e.g. some power-law fitting timeframes had activity of more than 50% of maximum network activity and some power-law fitting timeframes had activity below that 50% mark. This behaviour is reinforced, when the networks are cut using sliding-windows. In Figure 4.6, 20 timeframes, fitting to a power-law and therefore evolving based on the preferential attachment model, have activity less than 50% of the maximum projects commit-activity, there are also seven timeframes, matching the preferential attachment evolution model, with more than 50% activity.

For the mail-based network, most power-law fitting timeframes have activity less than 25% of the maximum projects mail-activity. When we look at the sliding-window cut network with a timeframe length of three months[2], only one of the 35 power-law fitting timeframes has activity greater than 25% of the maximum network activity (~32%). This configuration is chosen for visualization, because it provides more timeframes than any other configuration, which results in a more precise picture for a correlation between preferential attachment and project activity.

OPENSSL Regarding the file-based network of OpenSSL, in all configurations, most power-law fitting timeframes had activity of less than 50% of the maximum activity of its network. The only two configurations, where a power-law fitting timeframe has activity of more than 50% of the maximum activity of its network are nine-month, sliding-window configuration with one of four power-law fitting timeframes being over the 50% mark, and the nine-month, sliding-window configuration with one of 20 power-law fitting timeframes being over the 50% mark.

2 See Figure 4.7

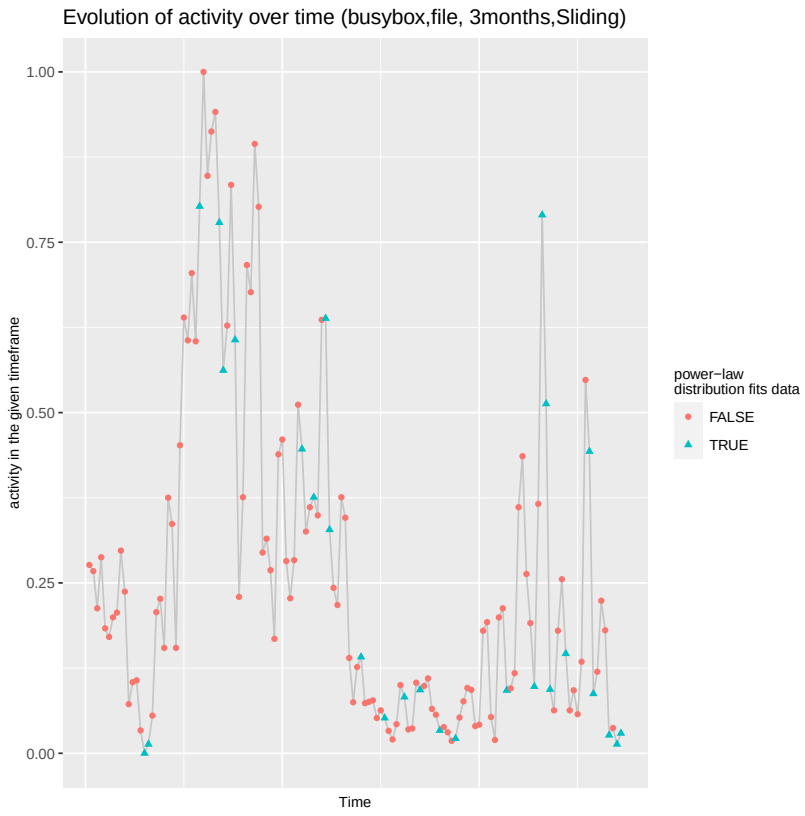


Figure 4.6: The graph for comparison between preferential attachment and commit activity for the three-month, sliding-window split, file-based network for Busybox.

For the mail-based network of OpenSSL, again most power-law fitting timeframes in all configurations have activity less then 50% of the maximum activity of its network, with the exception of the three-month, sliding-window configuration, where one of 60 power-law fitting timeframes having a greater percentized activity than 50% of the maximum network activity with ~63% of the maximum network activitiy, as seen in Figure 4.8.

QEMU For the file-based network of QEMU, no real statement can be made due to the absence of enough power-law fitting timeframes, e.g. in the three-month, non-sliding configuration, only one power-law fitting timeframe is present, having ~20% of the networks maximum activitiy, while in the nine-month, non-sliding configuration, two power-law fitting timeframes are present with one having ~10% of the networks maximum activity and the other having ~62% of the networks maximum activity. When the results of the configurations with a sliding-window approach are observed, this behaviour does not change, except that the total amount of non-power-law fitting timeframes increases.

For the mail-based network of QEMU, there is only one configuration having power-law fitting timeframes, namely the three-month, sliding-window configuration. In this configuration, only two timeframes seem to follow the preferential attachment model, being about 2% of the total timeframes in this configuration. These timeframes have a percentized activitiy of about 7% and 28% of the maximum of the networks activity.

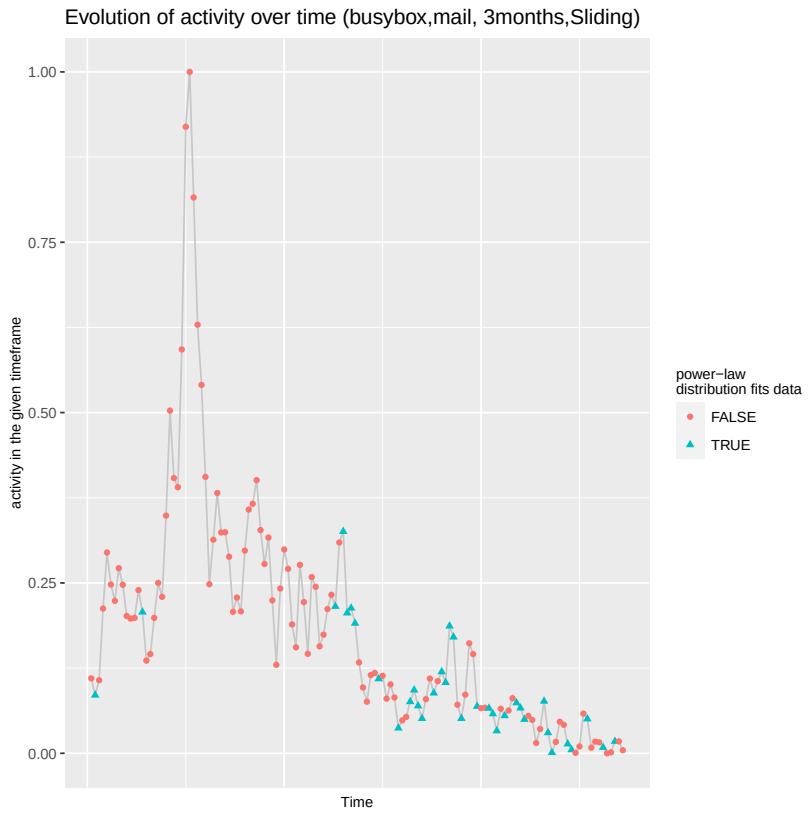


Figure 4.7: The graph for comparison between preferential attachment and mailing-list activity for the three-month, sliding-window split, file-based network for Busybox.

4.1.4 *RQ4: What impact do different timeframe lengths have on analysis results?*

The tables that are used for the comparison of timeframe lengths have columns for each timeframe length, namely three months, six months and nine months. The values in the bottom row of each table are the percentages of timeframes following the preferential attachment model in regard to the total amount of timeframes as well as the absolute number of timeframes following the preferential attachment model (written in braces after the percentage).

BUSYBOX For the file-based network of Busybox, the highest percentage of power-law fitting timeframes is found with the nine-month timeframe length (~13% with three fits), the second highest percentage is found with the three-month timeframe length (~10% with seven fits) and the lowest percentage is found with the six-month timeframe length (~8% with three fits).

Including the sliding-window approach, the highest percentage of power-law fitting timeframes can again be found with the nine-month timeframe length (~25% with eleven fits), three-month and six-month timeframe lengths produce less power-law fitting timeframes (~19% with 27 and 13 fits, respectively).

For the mail-based network of Busybox, the highest percentage of power-law fitting timeframes is found with both the three-month and the six-month timeframe lengths (~5% with

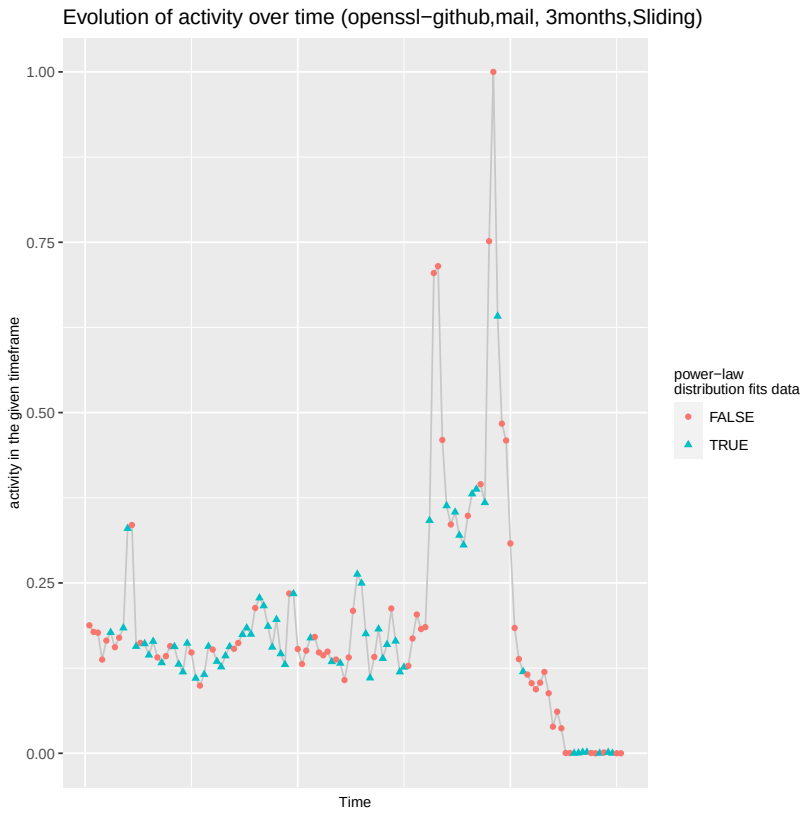


Figure 4.8: The graph for comparison between preferential attachment and mailing-list activity for the three-month, sliding-window split, mail-based network for OpenSSL.

four and two fits, respectively). For the nine-month timeframe length, not a single timeframe did fit a power-law.

When a sliding-window approach is applied, the highest proportion of power-law fitting timeframes is returned when using the three-month timeframe length (~25% with 35 fits), the second highest proportion is found using the six-month timeframe length (~10% with seven fits) and the lowest percentage of power-law fitting timeframes is found using the nine-month timeframe length (~9% with four fits).

This data can be found in a tabluar form in 4.4 for the non-sliding-window approach and in 4.5 for the sliding-window approach.

Table 4.4: Proportion of power-law fits for Busybox: 4.5a shows the proportion of power-law fits for the file-based network and 4.5b shows the proportion of power-law fits for the mail-based network.

File

3 months	6 months	9 months
10% (7)	8% (3)	13% (3)

(a)

Mail

3 months	6 months	9 months
5% (4)	5% (2)	0% (0)

(b)

Table 4.5: Proportion of power-law fits for Busybox: 4.6a shows the proportion of power-law fits for the file-based, sliding-window split network and 4.6b shows the proportion of power-law fits for the mail-based, sliding-window split network.

File (sliding-window)		
3 months	6 months	9 months
19% (27)	19% (13)	25% (11)

(a)

Mail (sliding-window)		
3 months	6 months	9 months
25% (35)	10% (7)	9% (4)

(b)

OPENSSL When looking at the non-sliding, file-based network of OpenSSL, the highest proportion of power-law fitting timeframes is found within the nine-month timeframe length (~8% with two fits), the second highest proportion of power-law fitting timeframes is found using the three-month timeframe length (~6% with five fits) and no power-law fitting timeframe is found using the six-month timeframe.

When applying the sliding-window approach, the highest proportion of power-law fitting timeframes is found within the three-month timeframe length (~13% with 20 fits), the second highest proportion of power-law fitting timeframes is found using the six-month timeframe length (~9% with seven fits) and the lowest proportion of power-law fitting timeframes is found within the nine-month timeframe length (~8% with four fits).

For the mail-based network of OpenSSL, the highest proportion of power-law fitting timeframes is found within the three-month timeframe length (~15% with ten fits), the second highest proportion of power-law fitting timeframes is found using the six-month timeframe length (~6% with two fits) and the lowest proportion of power-law fitting timeframes is found within the nine-month timeframe length (~4% with one fit).

When applying the sliding-window approach, the highest proportion of power-law fitting timeframes is found within the three-month timeframe length (~47% with 60 fits), the second highest proportion of power-law fitting timeframes is found using the six-month timeframe length (~27% with 17 fits) and the lowest proportion of power-law fitting timeframes is found within the nine-month timeframe length (~19% with eight fits).

This data can be found in a tabluar form in 4.6 for the non-sliding-window approach and in 4.7 for the sliding-window approach.

Table 4.6: Proportion of power-law fits for OpenSSL: 4.7a shows the proportion of power-law fits for the file-based network and 4.7b shows the proportion of power-law fits for the mail-based network.

File		
3 months	6 months	9 months
6% (5)	0% (0)	8% (2)

(a)

Mail		
3 months	6 months	9 months
15% (10)	6% (2)	4% (1)

(b)

QEMU When looking at the non-sliding, file-based network of QEMU, the highest proportion of power-law fitting timeframes is found within the nine-month timeframe length (~9%

Table 4.7: Proportion of power-law fits for OpenSSL: 4.8a shows the proportion of power-law fits for the file-based, sliding-window split network and 4.8b shows the proportion of power-law fits for the mail-based, sliding-window split network.

File (sliding-window)

3 months	6 months	9 months
13% (20)	9% (7)	8% (4)

(a)

Mail (sliding-window)

3 months	6 months	9 months
47% (60)	27% (17)	19% (8)

(b)

with two fits), the second highest proportion of power-law fitting timeframes is found using the six-month timeframe length (~5% with two fits) and the lowest proportion of power-law fitting timeframes is found within the three-month timeframe length (~1% with one fit).

When applying the sliding-window approach, the highest proportion of power-law fitting timeframes is found again within the nine-month timeframe length (~4% with two fits), the second highest proportion of power-law fitting timeframes is found using the three-month timeframe length (~3% with five fits) and the lowest proportion of power-law fitting timeframes is found within the six-month timeframe length (~2% with two fits).

For the mail-based network of QEMU, without using sliding-windows, there are no power-law fitting timeframes, therefore each timeframe length gives the same result in this configuration.

When applying the sliding-window approach, power-law fitting timeframes can only be found in the three-month timeframe length (~1% with two fits). The six-month timeframe length as well as the nine-month timeframe length do not produce any power-law fitting timeframes.

This data can be found in a tabluar form in 4.8 for the non-sliding-window approach and in 4.9 for the sliding-window approach.

Table 4.8: Proportion of power-law fits for QEMU: 4.9a shows the proportion of power-law fits for the file-based network and 4.9b shows the proportion of power-law fits for the mail-based network.

File

3 months	6 months	9 months
1% (1)	5% (2)	9% (2)

(a)

Mail

3 months	6 months	9 months
0% (0)	0% (0)	0% (0)

(b)

When trying to find out similarities between node-based splitting and edge-based splitting, not only the total-edge network should be taken into consideration, but also the internal-edge and external-edge networks for each project. Using tables, the results then can be compared to each other easily. In context of this analysis, matches means two timeframes, covering the same time period, one being edge-based and the other being node-based, both evolve based on preferential attachment or both do not evolve based on preferential attachment.

Table 4.9: Proportion of power-law fits for QEMU: 4.10a shows the proportion of power-law fits for the file-based, sliding-window split network and 4.10b shows the proportion of power-law fits for the mail-based, sliding-window split network.

File (sliding-window)		
3 months	6 months	9 months
3% (5)	2% (2)	4% (2)

(a)

Mail (sliding-window)		
3 months	6 months	9 months
1% (2)	0% (0)	0% (0)

(b)

BUSYBOX For the file-based networks of Busybox, in the three-month, total-edge networks, 45% of the timeframes match, for the internal-edge networks 54% of the timeframes match and for the external-edge networks 72% of the timeframes match.

For the six-month timeframe length, 72% of the timeframes match for both the total-edge networks and the internal-edge networks. For the external edge-networks, 81% of the timeframes match.

Looking at the nine-month timeframe length, for the total-edge networks 63% of the timeframes match, for the internal-edge networks 54% of the timeframes match and for the external-edge networks 72% of its timeframes match.

For the mail-based networks, the overall picture looks a bit different: for all three timeframe lengths, the total-edge networks as well as the external-edge networks match 100%, for the internal-edge networks for all timeframe lengths, all networks match 63%.

A table containing the results for Busybox can be found in 4.10.

Table 4.10: Comparison between splitting types for Busybox (non-sliding). Each row represents a timeframe length, each column represents the edge-type that is compared. Values in the cells represent the similarity between the edge-based and the node-based network for the associated timeframe length and edge-type.

	Total	Internal	External
three-months	45%	54%	72%
six-months	72%	72%	81%
nine-months	63%	54%	72%

(a) File based

	Total	Internal	External
three-months	100%	63%	100%
six-months	100%	63%	100%
nine-months	100%	63%	100%

(b) Mail based

OPENSSL Regarding the file-based networks of OpenSSL, the results are similar to those of Busybox, with the three-month, total-edge networks also having 45% matching timeframes, 63% matching timeframes for the internal-edge networks and also 72% matching timeframes in the external-edge networks.

For the six-month timeframe length, both total-edge and external-edge networks have 45% in matching timeframes for the internal-edge networks, 54% of the timeframes match.

For the nine-month timeframe length, 36% of the timeframes match for the total-edge networks, for the internal-edge networks 45% of the timeframes match and for the external-edeg timeframe length, 63% of the timeframes match.

Moving on to the mail-based networks for OpenSSL, for the three-month timeframe length, 90% of the timeframes from the total-edge networks match, 45% of the internal-edge networks match and 72% of the external-edge networks match.

For the six-month timeframe length, for the total-edge networks 81% of the timeframes match, for the internal networks 27% of the timeframes match and for the external-edge networks, 81% of the timeframes match.

Looking at the nine-month timeframe lengths, 100% of the timeframes for both the total-edge networks as well as the external-edge networks match. For the internal-edge networks 45% of the timeframes match.

A table containing the results for Busybox can be found in 4.11.

Table 4.11: Comparison between splitting types for OpenSSL (non-sliding). Each row represents a timeframe length, each column represents the edge-type that is compared. Values in the cells represent the similarity between the edge-based and the node-based network for the associated timeframe length and edge-type.

	Total	Internal	External
three-months	45%	63%	72%
six-months	45%	54%	45%
nine-months	36%	45%	63%

(a) File based

	Total	Internal	External
three-months	90%	45%	72%
six-months	81%	27%	81%
nine-months	100%	45%	100%

(b) Mail based

QEMU For QEMU, there is not much variation. Looking at the file-based networks, in the three-month timeframe length, 100% of timeframes for all three edge-type networks match.

For the six-month timeframe length as well as for the nine-month, 81% of the timeframes match for the total-edge and internal-edge networks. For the external-edge networks, 90% of the timeframes match.

For the mail-based networks, when looking at the three-month timeframe length, 100% of timeframes match for both the total-edge and external-edge networks. For the internal-edge networks, 72% of the timeframes match.

Regarding the six-month timeframe length, again for both the total-edge and external-edge networks, 100% of the timeframes match. For the internal-edge networks, 90% of the timeframes match.

For the nine-month timeframe lenght, for all networks for all edge-types, 100% of the timeframes match.

A table containing the results for Busybox can be found in 4.12.

4.2 DISCUSSION

Table 4.12: Comparison between splitting types for QEMU (non-sliding). Each row represents a timeframe length, each column represents the edge-type that is compared. Values in the cells represent the similarity between the edge-based and the node-based network for the associated timeframe length and edge-type.

	Total	Internal	External
three-months	100%	100%	100%
six-months	81%	81%	90%
nine-months	81%	81%	90%

(a) File based

	Total	Internal	External
three-months	100%	72%	100%
six-months	100%	90%	100%
nine-months	100%	100%	100%

(b) Mail based

4.2.1 *RQ1: How do new authors connect to a developer network?*

In all three projects, the amount of new-to-new connections in their file-based networks is vanishingly small, so there seems to be nearly no interaction between newcomers on a commit basis. Newcomers seem to preferably work on files mostly previously active developers created or edited in the past, forming external edges in the networks. These external edges however also only make up a small proportion of all new edges in the file-based networks, which means that most edges are created by developers who have been previously active already. An interesting observation regarding external edges, is that while for Busybox and OpenSSL, the proportions of external edges in regards to the total amount of edges seems to be similar to each other, QEMU, which has about ten times more edges in its file-based network than the aforementioned two projects, has a way smaller proportion of external edges compared to the internal edges.

When mail-based networks are inspected, it is directly clear that both networks differ in terms of proportions of edge types. The majority of edges in all three projects networks is made up by external edges with more than 50% of all edges most of the time, which can have multiple reasons, some being that new developers tend to ask questions which are then mostly answered by already active developers or newcomers saying "thank you" to a developer who fixed or wrote about an issue they also have, which were encountered when reading samples from the associated mailing-lists. New-to-new edges are also more present than in the file-based networks, meaning that new developers might interact with each other directly or they interact with an already active developer in the same mailing-list thread.

4.2.2 *RQ2: Do these projects evolve based on preferential attachment?*

Looking at the networks with all types of edges (total edges) of all three projects, no project seems to really evolve based on preferential attachment. With a proportion of power-law fitting timeframes of 0% to ~15% for both mail-based and file-based networks, depending on the chosen timeframe length, the majority of the timeframes degree distributions of all three projects does not fit a power-law distribution and therefore does not evolve based on preferential attachment. This behaviour does change however when only internal edges are taken into consideration; for both, Busybox and OpenSSL, the proportion of power-

law fitting timeframes in their mail-based networks is substantially higher, proportions of 59% to 88% of power-law fitting timeframes. Since the majority of timeframes for these two projects does evolve based on the preferential attachment model, it can be concluded, that for both projects their mail-based networks tend to follow the preferential attachment model. Based on personally reading some threads on these mailing-lists, an explanation for this phenomenon would be that it seems that developers can be categorized into two groups: one being developers who only write messages in a few mailing-list threads, the other being developers who tend to answer questions in a wider variety of threads. Those developers that fall into the second group can therefore create connections to a potentially bigger set of developers than those who only connect to other people in a limited set of threads. For OpenSSL, this behaviour might have changed, since the projects communication did switch from a mailing-list environment to the Github issues page[3].

For Busybox, the proportion of preferential-attachment-evolving timeframes for internal edges also increases slightly for their file-based networks, seeing an increase of ~30%, so there are some tendencies for preferential attachment on a file-basis, although most timeframes do not evolve based on preferential attachment.

When looking at the networks for external edges, the proportions of preferential-attachment-evolving timeframes for QEMU for its file-based network sees a huge increase from a maximum of 9% in the total network to a maximum of 35% of power-law fitting timeframes. This still means that a minority of timeframes evolve based on preferential attachment, but it shows, that nearly the only tendencies of preferential attachment evolution can be found in the external, file-based network. Based on a look at commits from new developers and the contribution FAQ, most new developers mostly submit patches and fixes for reported bugs. Since most of this work takes place inside already existing files, they immediately connect to all other people who edited those files. With some developers having more than 3000 commits, chances are high that a new developer connects to one of those core developers by editing a file.

As literature states, that network evolution based on preferential attachment leads to scale-free networks[15], a quick look at the scale-freeness property of each timeframe has shown, that most timeframes, taken as seperate networks, are scale-free, even for network configurations where the majority of timeframes does not evolve based on preferential attachment.

4.2.3 *RQ3: Is there a relation between preferential attachment and the amount of activity in the project?*

For the file-based networks of all three projects, no majority of timeframes did evolve based on preferential attachment. For finding out a correlation between the number of commits and the evolution of preferential attachment, for most configurations there were way too few timeframes that did actually evolve based on preferential attachment. For those configurations, mainly sliding-window cut network configurations, that had a higher proportion of preferential attachment evolving timeframes, there was no clear correlation between activity. Only based on this little amount of data, one might say, that when the number of commits is low, the chance of the network evolving based on preferential attachment is increased.

3 https://github.com/openssl/openssl/issues

When looking at the mail-based networks, the proportion of power-law fitting timeframes does increase with less mail activity, e.g. OpenSSL having most of its timeframes in the three-month, sliding window configuration following the preferential attachment model, when activity was less than 25% of the maximum projects activity. A reason for this could be that, core developers are communicating most of the time with other active developers. During times, a new security security vulnerability was found, lots of developers started to use the mailing list in order to find out information or simply try to contribute with information. This would result in more connections forming between non-core developers, leading to a higher rate of acquiring new links for those developers who were not that highly connected before, leading to a non-preferential attachment based network evolution.

As a general assertion it might be, that the forming of preferential attachment based evolution in a network is supported by a low project activity, most prominently for mail-based networks. Reasons for QEMU not behaving like this for most configurations might be that the number of mails and commits is generally higher than in the other two projects.

4.2.4 *RQ4: What impact do different timeframe lengths have on analysis results?*

When comparing the proportion of power-law fitting timeframes for all projects, no pattern can be detected. There are situations, e.g. when comparing the file-based, three-month configuration for Busybox with its file-based, nine-month configuration, where a power-law fitting timeframe in the nine-month configuration can be mapped onto one or more power-law fitting timeframes in the three-month configuration (e.g. timeframes seven to nine in the three-month configuration onto timeframe three of the nine-month configuration). Mappings like this can be seen as a noise-reducing summary, where flucuations in the fine-grained, three-month configuration are aggregated, resulting in a more generalized classification, i.e. only saying whether or not nine months in this network are evolving based on the preferential attachment model instead of having confusing fluctuations where whether or not three months evolving based on preferential attachment can alternate. However, situations like this are rare, leaving differences in results between different timeframe lengths, where there are timeframes evolving based on preferential attachment for one timeframe-length configuration, where there is no timeframe evolving based on preferential attachment during the same time (and the other way around).

Since there are differences in results between different timeframe lengths, based on the findings for this thesis, no statement about the cause and the impact of different timeframe lengths can be made.

Regarding edge-type comparison, some very interesting patterns emerge. Since for QEMU, the amount of preferential attachment evolving timeframes is negligible low, most timeframes do match for both file-based networks and mail-based networks. For Busybox and its mail-based networks, the timeframe length does not make a difference in edge-type comparison; it only differs between the internal-edge type and both the total-edge and external-edge type, both matching timeframes at 100%. When we look at the proportion of power-law fitting timeframes for the mail-based networks of Busybox, both total-edge and external-edge networks have nearly no preferential attachment evolving timeframes unlike the internal-edge networks. A similar, but more varying picture can be seen for OpenSSL. Those network con-

figurations that have a higher proportion of preferential attachment evolving timeframes have less matching timeframes when comparing edge-based splitting with node-based splitting.

Based on the three projects, it seems that the relation between edge-based splitting and node-based splitting can be generalized as follows: The higher the proportion of preferential attachment evolving timeframes, the lower the amount of matching timeframes when comparing edge-based splitting with node-based splitting.

4.3 THREATS TO VALIDITY

In this section, we will discuss possible threats to validity. Possible threats to validity that are being discussed are both internal threats to validity as well as external threats to validity.

4.3.1 *Internal Threats to Validity*

Timeframe lengths have to be relatively short[13], but no optimal length is defined. Barabási used timeframes of the length of one year, so the chosen timeframe lengths of three, six and nine months should be fine. It is also mentioned, that, if a network develops a stationary state, the degree distribution becomes independent of the points in time, the timeframes are taken[13], but there is no definition of said stationary state. This should not cause any problems, because when the points in time that were chosen do not affect the degree distribution, using these points to calculate the results would also not have any impact on the degree distribution.

There is no best-practice or proven way for choosing the best start time for network splitting. It is done in a rather arbitrary way. Even though, network splitting was also done using the sliding-window approach, this only gives a slightly better resolution, i.e. there are alternative timeframes with a slightly different (half of the timeframe length) cutting point for not-so-perfectly cut timeframes.

4.3.2 *External Threats to Validity*

All projects for this thesis are comparably older projects, all having in common that they are built upon mailing-list based communication. Since the term open-source software project does not only include projects with mailing lists but also projects hosted on, e.g. Github, where communication takes mostly place on forum-like issue threads, findings about mail communication in this thesis might not be mappable onto those newer projects having issue boards instead of mailing lists.

Also, since projects on Github are easier to access for new developers due to this platforms features and sheer amount of hosted projects, the inital hurdle of joining such a project might be lower than for older projects, that are hosted on their own servers without browser-based inspection tools for files and code.

Another threat is the age of the analysed projects: while those projects have around twenty years of development, findings for these projects might not be mappable onto newer projects having just a few years of development.

Also, since there are already differences between the analysed projects, such differences can also occure between other projects, that were not analysed in the scope of this thesis.

5 RELATED WORK

This chapter covers related work for the preferential attachment field as well as similar topics.

A general analysis for measuring preferential attachment for a macroscopic scope was conducted by H Jeong, Z Néda, and A. L Barabási[13] for *a*) coauthorship networks of neuroscience, *b*) citation networks, where each node represents a published paper and each edge represents a citation of said paper, *c*) actor networks, where nodes represent actors and edges represents the collaboration between two actors in the same movie and *d*) internet data, where autonomous systems were mapped to nodes and direct connections between those systems created edges between those nodes. This work focussed on retrieving the exponent of the power-law fitting for each network in order to classify the networks as either sub-linear preferential attachment, linear preferential attachment or super-linear preferential attachment[13].

In 2006, A. Capocci et al.[7] conducted an analysis for the growth of the online encyclopedia Wikipedia. The authors used articles as vertices and hyperlinks between articles as directed edges, resulting in a directed network. They found out, that the article network of Wikipedia was growing based on the preferential attachment model. Due to its age, this work might be outdated by today. Additionally, the topology of the network that was used for the analysis, based on Wikipedia, could be different from the topologies of the developer-community networks of the open-source software projects that were used for this thesis.

Regarding open-source software projects, Wonseok Oh and Sangyong Jeon[18] released a paper in 2007 covering membership dynamics and network characteristics of such open-source software projects and how the stability of a developer network is impacted by these dynamics and characteristics. The authors found out, that when external influences, e.g. "the availability of other OSS projects"[18], were weak, the developer networks were rather random than scale-free[18].

Studies about the organizational structure of 18 open-source software projects was conducted by Mitchell Joblin, Sven Apel and Wolfgang Mauerer[15]. A network-analytical approach was applied in order to find out, how open-source projects manage themselves. This study gave some very interesting results, being that developers *a*) seem to form networks which have the scale-free property, *b*) gather more and more requirements for coordination as the project grows and *c*) they are arranged hierarchically initially, but when the project starts to grow in terms of developers, only the core team seems to keep this hierarchical structure. All things considered, it seems like open-source software projects do not follow "conventional software-engineering wisdom"[15].

Organization of teams of open-source software projects was also examined by Kevin Crowston et al.[9]. Based on three active open-source software projects, the authors analysed self-organization in those projects, task assignment to be more precise. Their core finding was, that self-assignment of tasks was the most common task-assignment method, so most developers chose the work they wanted to do themselves[9].

Other work in regards of collaboration networks between developers, an approach for network mining and visualization from version control systems, is presented by Andrejs Jermakovics et al.[14]. With their techniques, the authors were able to compute similarities between developers based on file-edits, by assigning weights onto files and using a cosine measure to compute developer similarity. To improve the modularity of their approach, the authors have included filtering techniques[14].

Unlike the previously mentioned research projects, that focussed evolution on a network basis, an empirical study was conducted by Guowu Xie and Jianbo Chen and Iulian Neamtiu[23] for the software evolution of open-source software. Seven open-source software projects, including OpenSSH, were used in order to analyse each of their official releases. The authors of the study tried to verify the laws of software evolution by Lehman and Belady. They concluded that half of the laws, namely Continuing Change, Increasing Complexity, Self Regulation and Continuing Growth do still apply for open-source projects, whereas Conservation of Organizational Stability, Conservation of Familiarity, Declining Quality and Feedback System could not be validated for the chosen projects. They also found out, that different branches of open-source software does evolve in parallel and that all investigated projects have so called "change hot spots", meaning that "a high percentage of changes are concentrated to a small percentage of code"[23].

Regarding the onboarding process for open-source software projects, also addressed in this thesis, a case study by Fabian Fagerholm et al.[11] was conducted. They found out that core developers spending some time on mentoring as well as organizing events (the authors give the example of hackathons) can fuel the onboarding process[11].

Further work in direction of onboarding guidelines was done by Igor Steinmacher et al.[22]. Unlike the studies conducted by Fabian Fagerholm[11], does not take company-backed open-source projects into account, but focusses on mainly community-backed projects. The article introduces guidelines for both, open-source communities that want more contributions by external developers as well as for newcomers who want to start contributing to open-source projects. These guidelines are split into community guidelines and newcomer guidelines, which again are split into social as well as technical guidelines. For the community guidelines, the authors propose implementing newcomer-specific portals including contribution guidelines and readme files. Also newcomers should be assigned to tasks that are considered easy. When a newcomer asks questions about the project or a task they are assigned to, the request should be answered as quickly as possible, otherwise the newcomer, who most likely is a volunteer, loses their motivation for working for the project. The guidelines for newcomers are partially similar to the community guidelines, e.g. instead of getting assigned onto an easy task, it is proposed that a newcomer should try to find an easy task themselves, if they are not assigned onto a task by the community. A more technical suggestion for newcomers is setting up a virtual machine just for this single project in order to make dependency management easier[22]. All these guidelines can have an impact on how developers connect to an open-source community network.

6 CONCLUDING REMARKS

This chapter covers the final findings of this thesis as well as possible future work, that can be conducted based on these findings.

6.1 CONCLUSION

Regarding the connection of new authors to a developer network, it seems that newcomers connect more via mailing-lists than code commits. On mailing lists, most new connections between authors are created between newcomers and already active authors, while commit-based connections are mostly created between pre-active developers. Also, connections between newcomers, while extremely rare in the file-based networks, are common in all analysed mailing-list networks for all three projects .

Most of the time, all projects did not evolve based on preferential attachment. QEMU aside, the proportion of preferential attachment evolving timeframes is higher for mail-based networks than it is for file-based networks. For QEMU, there is only a noticeable number of preferential attachment evolving timeframes for the external-edge, file-based network.

When project activity is correlated with whether or not preferential attachment is present in a network of a project, it seems that the lower the project activity is, the higher is the proportion of preferential attachment evolving timeframes.

Timeframe lengths, on the other hand, seem to not have any distinct correlation with whether or not the network of a project evolves based on preferential attachment. The exact impact of timeframe length on preferential attachment analysis could not be determined since the results from different projects vary.

However for the splitting type, namely edge-based splitting or node-based splitting, there seems to be an inverse correlation between network splitting type and preferential attachment results, i.e. when the proportion of preferential attachment evolving timeframes is high for an edge-based network, the similarity between results from both splitting types was low as well as the other way around. This means, that for an edge-based network configuration that results in a high number of preferential attachment evolving timeframes, the associated node-based network configuration returns different results in terms of preferential attachment evolving timeframes. The other way around, for an edge-based network configuration that results in a low number of preferential attachment evolving timeframes, the associated node-based network configurations results, regarding timeframes that evolve based on preferential attachment, are similar to those of the edge-based configuration.

In summary, our results indicate that mailing-list based networks of smaller projects evolve based on preferential attachment for internal edges, during time periods of low volume of sent emails. For bigger projects, in this case QEMU, almost no timeframe evolved based on preferential attachment. For the file-based networks of all analysed projects, the amount of preferential attachment evolving timeframes was low.

6.2 FUTURE WORK

Based on this thesis, more modern open-source software projects can be analysed, e.g. React by Facebook. The example of React is also interesting to analyse because of its affiliation to a commercial company. Besides of the age of open-source software projects, projects that did not rely on mailing-list based communication but rather used issue-boards from the beginning could be interesting to investigate for preferential attachment and compare it to mailing-list based projects.

Regarding the scale-freeness property of a network, it might be interesting to get a better understanding about the relationship between the scale-freeness property and the preferential attachment evolution. Based on our results, an open-source software developer network does not need to evolve based on preferential attachment in order to result in a scale-free network. The question rises, what other reasons lead to the formation of scale-free networks in developer communities.

Also comparisons between different timeframe lengths did not yield any satisfying result. Since timeframe lengths do have a small impact on preferential attachment analysis, it might be interesing to find out how different timeframe lengths do affect the analysis in a more generalizable way.

Another interesting question would be what impact a preferential attachment evolving developer-network has on the associated software-project. Are there any assumptions that could be made about a software project based on whether or not its developer networks evolve based on preferential attachment?

APPENDIX

For reasons of space, only a few visualizations were shown in the main text of this thesis. In this chapter, you can find a collection of the used visualizations.

A.1 PLOTS

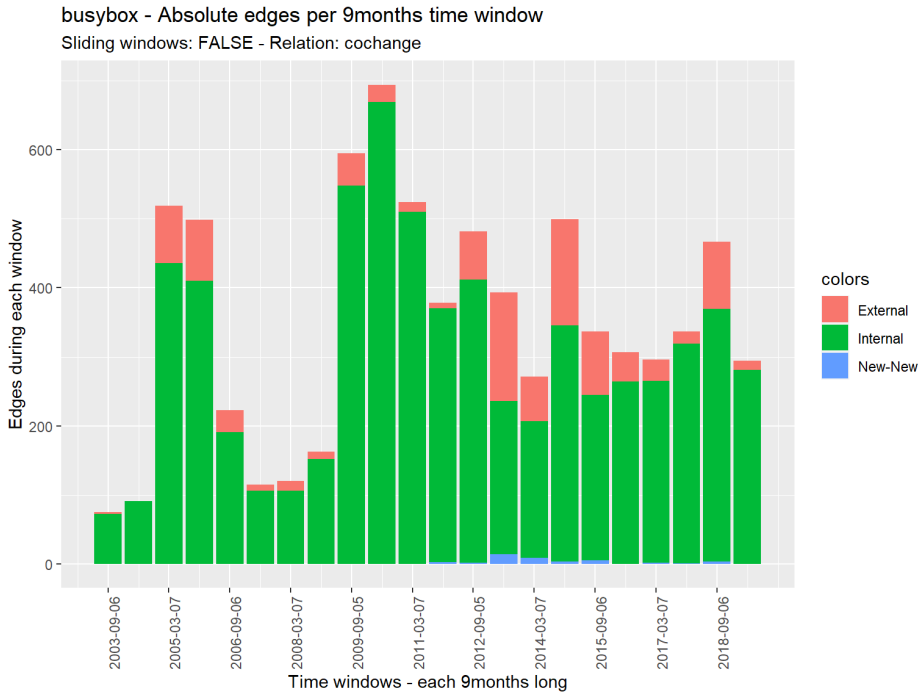


Figure A.1: The absolute creation rate for new edges for the commit-based network for the Busybox project with a nine-month timeframe length

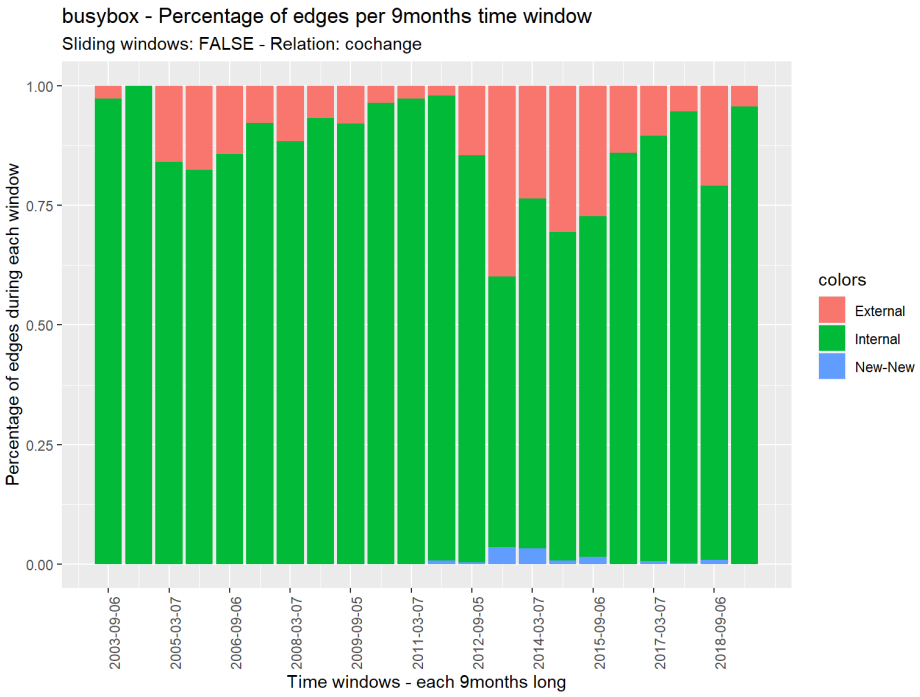


Figure A.2: The relative creation rate for new edges for the commit-based network for the Busybox project with a nine-month timeframe length

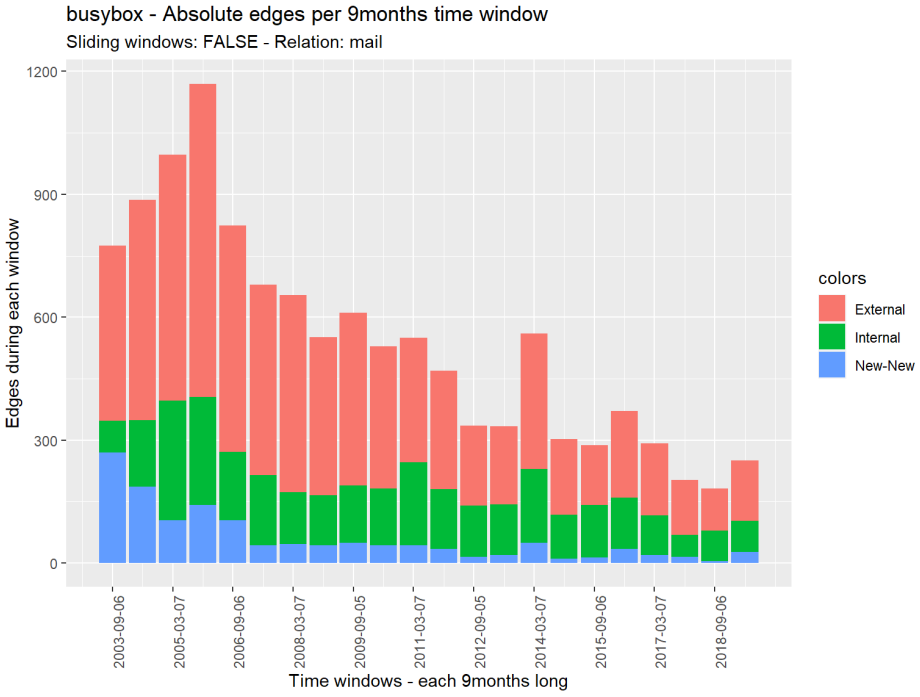


Figure A.3: The absolute creation rate for new edges for the mail-based network for the Busybox project with a nine-month timeframe length

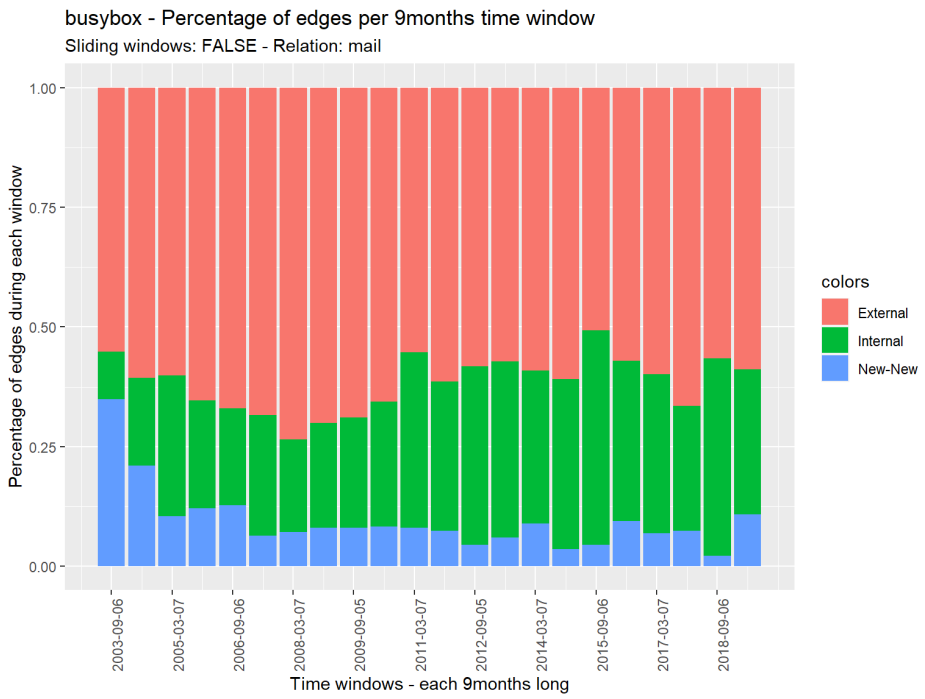


Figure A.4: The relative creation rate for new edges for the mail-based network for the Busybox project with a nine-month timeframe length

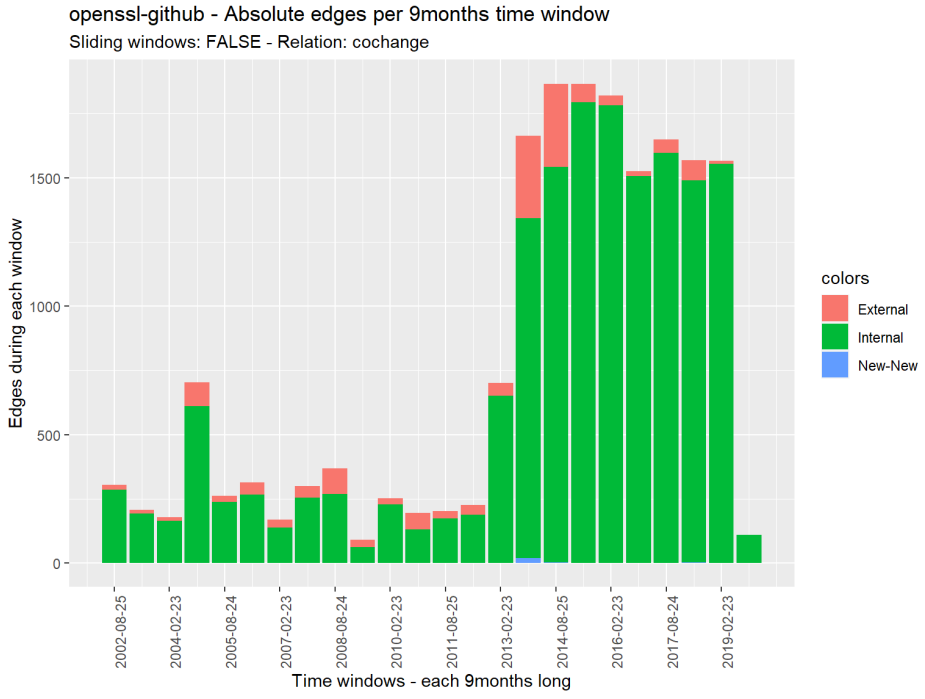


Figure A.5: The absolute creation rate for new edges for the commit-based network for the OpenSSL project with a nine-month timeframe length

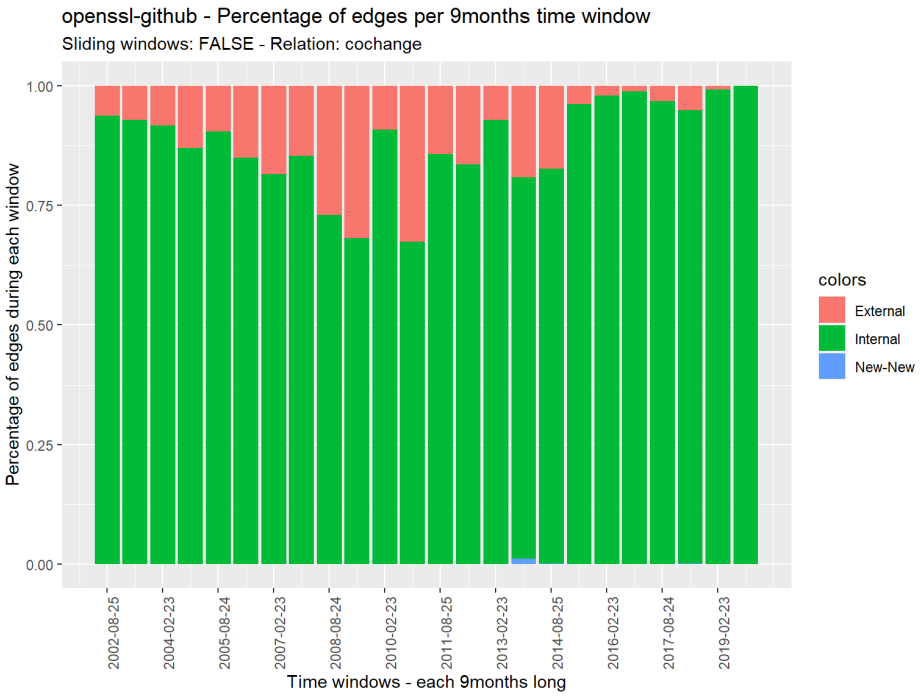


Figure A.6: The relative creation rate for new edges for the commit-based network for the OpenSSL project with a nine-month timeframe length

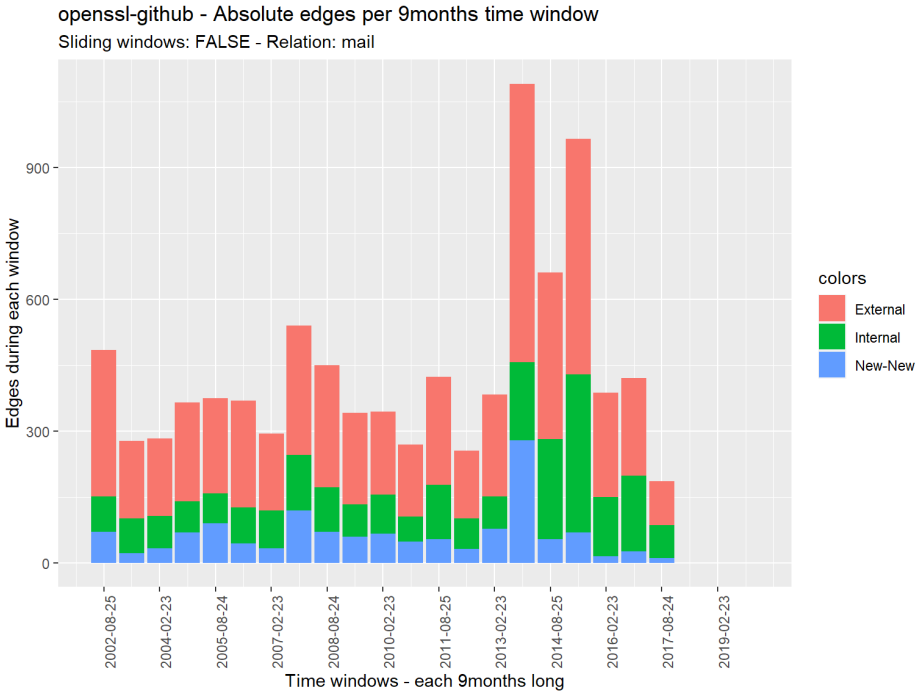


Figure A.7: The absolute creation rate for new edges for the mail-based network for the OpenSSL project with a nine-month timeframe length

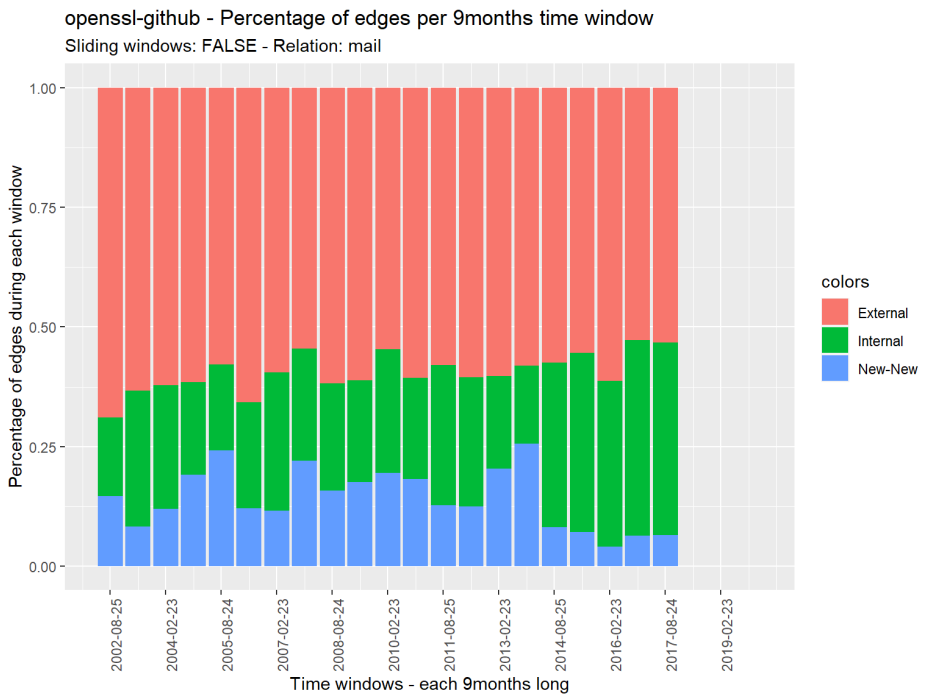


Figure A.8: The relative creation rate for new edges for the mail-based network for the OpenSSL project with a nine-month timeframe length

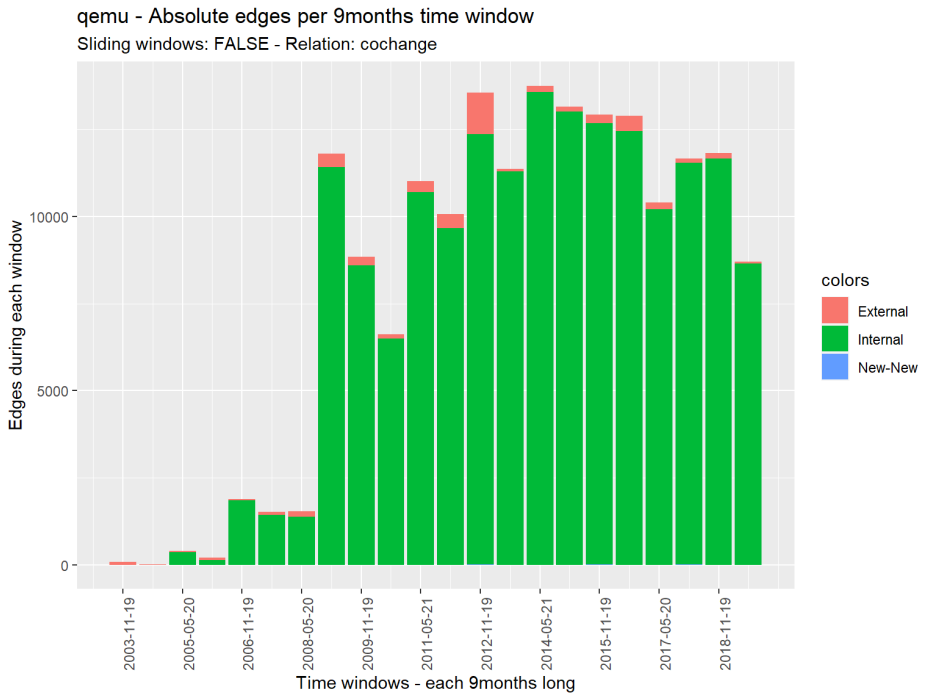


Figure A.9: The absolute creation rate for new edges for the commit-based network for the QEMU project with a nine-month timeframe length

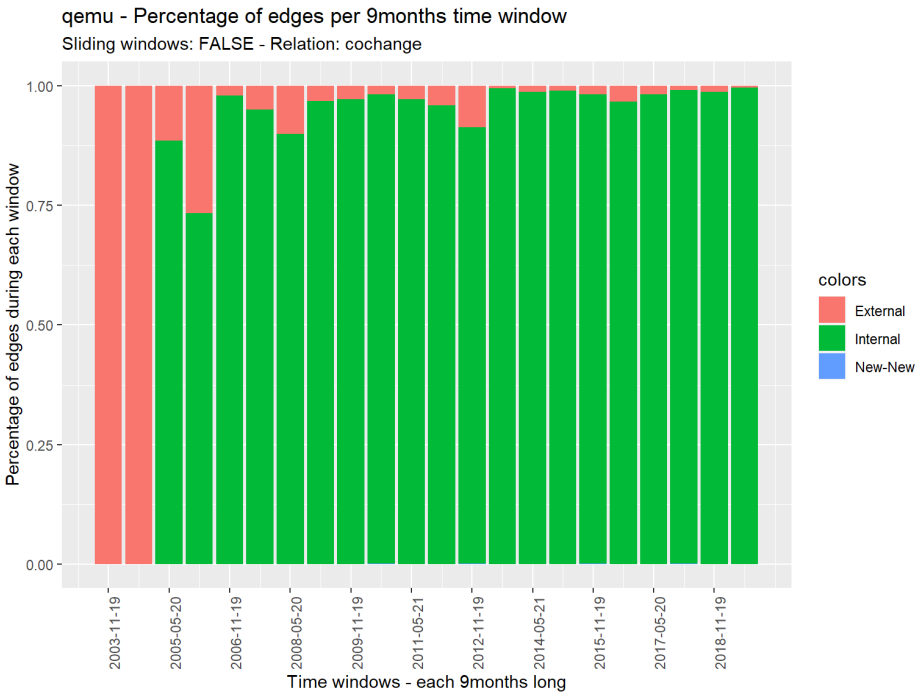


Figure A.10: The relative creation rate for new edges for the commit-based network for the QEMU project with a nine-month timeframe length

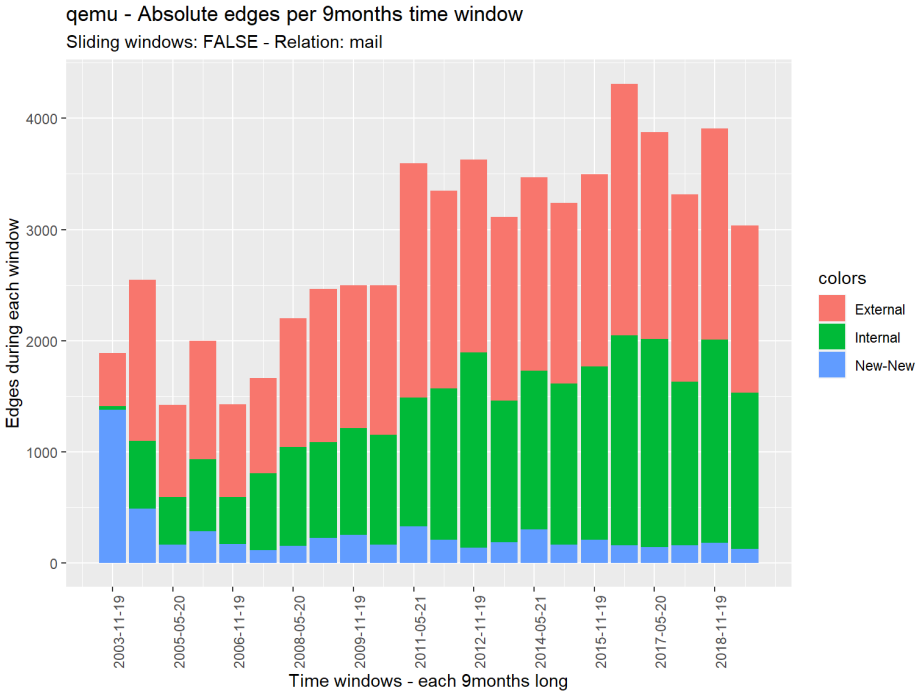


Figure A.11: The absolute creation rate for new edges for the mail-based network for the QEMU project with a nine-month timeframe length

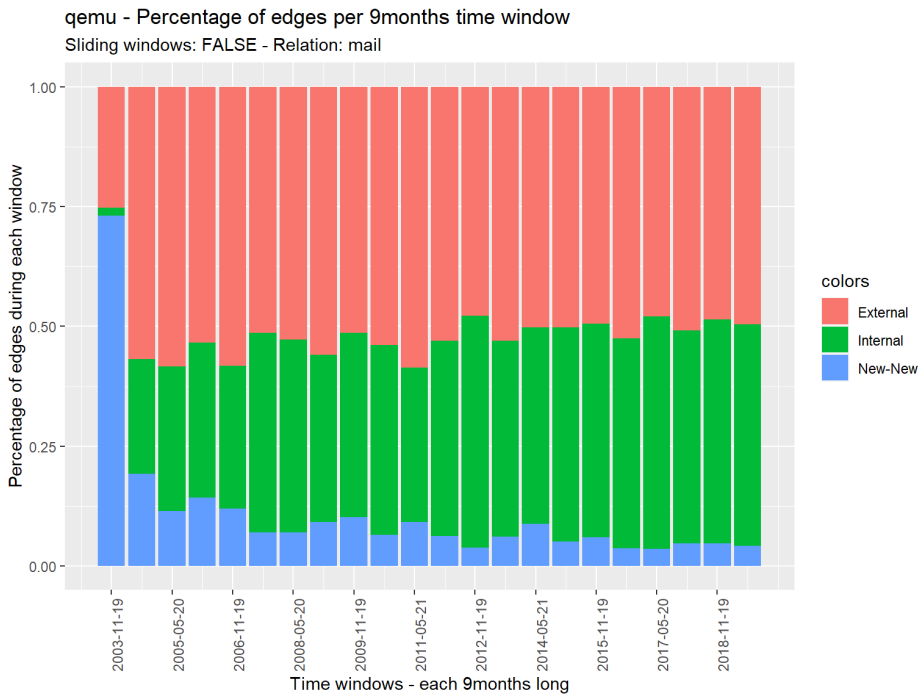
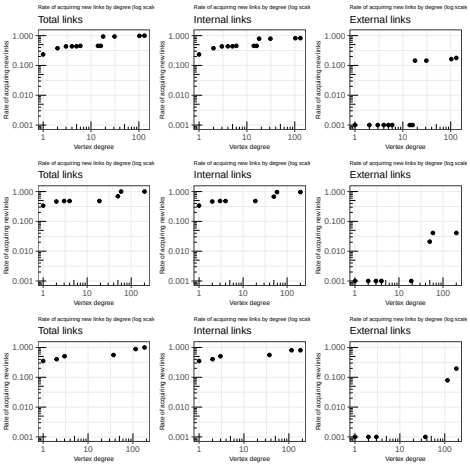
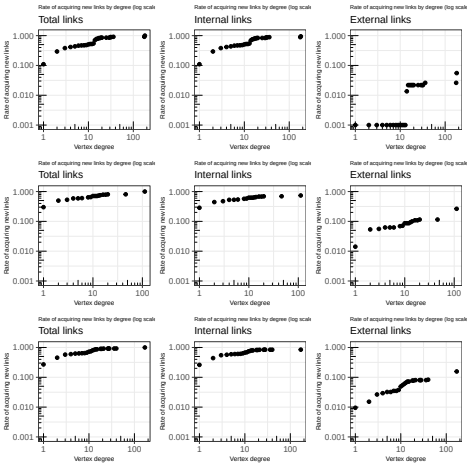


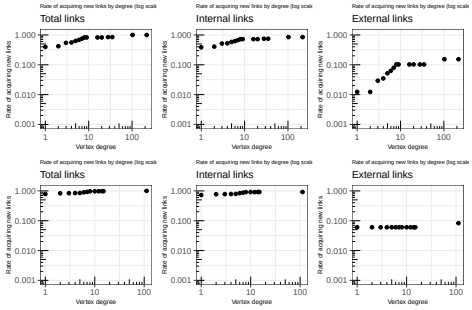
Figure A.12: The relative creation rate for new edges for the mail-based network for the QEMU project with a nine-month timeframe length



(a)

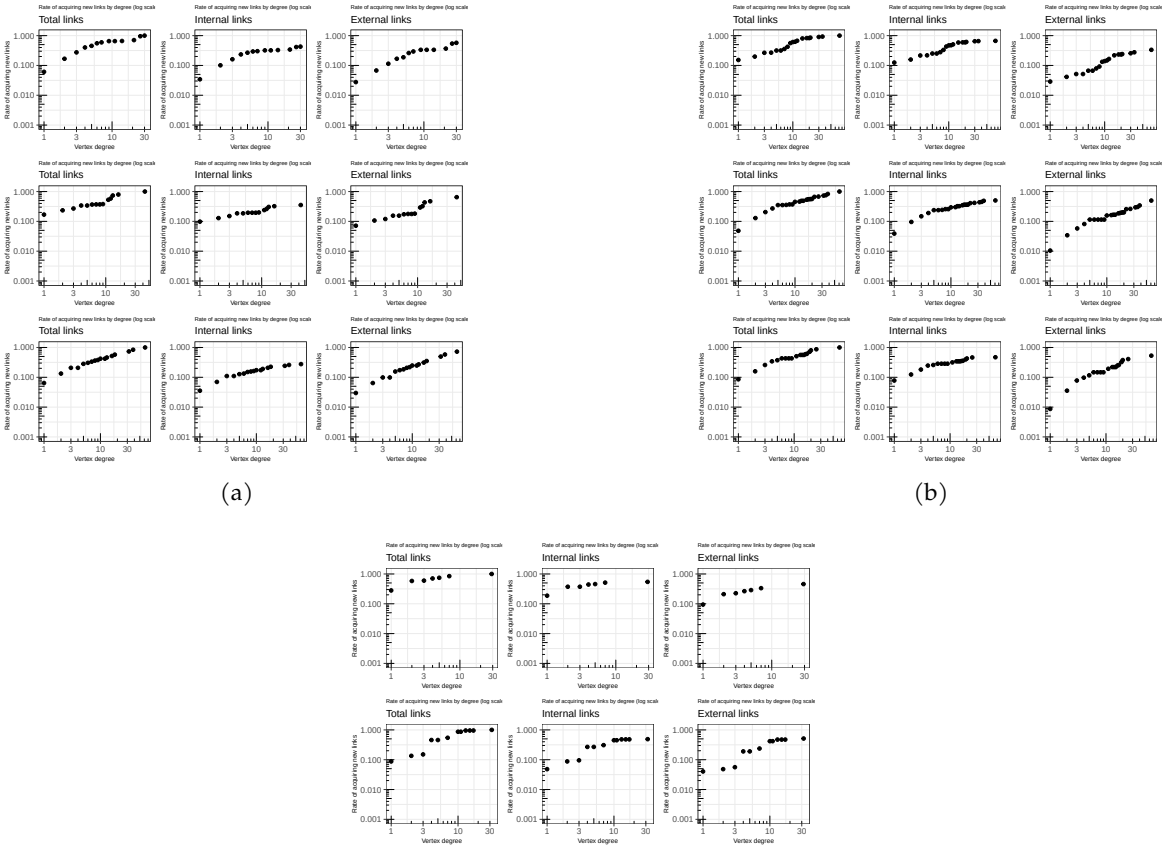


(b)



(c)

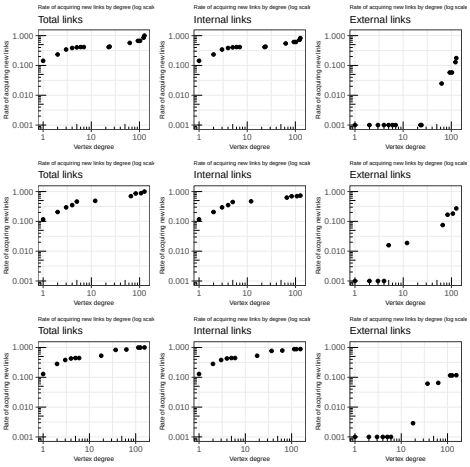
Figure A.13: The cumulative function for the three-month timeframe length of the commit-based data (edge-based split) for Busybox: In (a), visualizations for three three-month timeframes from 2003-03-08 until 2003-12-07 is shown. In (b), visualizations for three three-month timeframes from 2011-06-07 until 2012-03-07 is shown. In (c), visualizations for three three-month timeframes from 2019-09-06 end of record is shown.



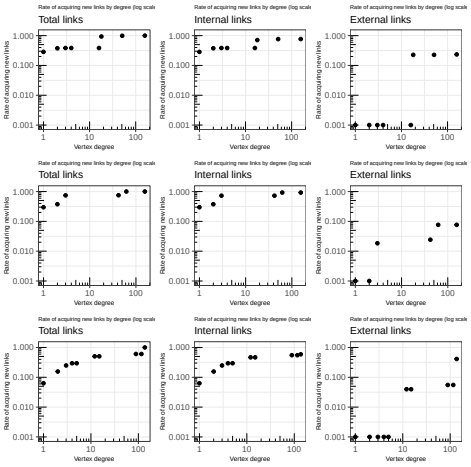
(a) (b)

(c)

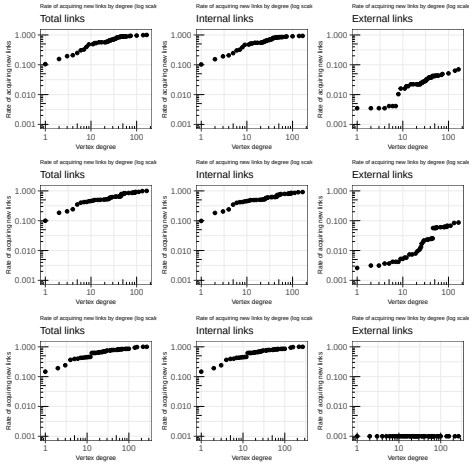
Figure A.14: The cumulative function for the three-month timeframe length of the mail-based data (edge-based split) for Busybox: In (a), visualizations for three three-month timeframes from 2003-03-08 until 2003-12-07 is shown. In (b), visualizations for three three-month timeframes from 2011-06-07 until 2012-03-07 is shown. In (c), visualizations for three three-month timeframes from 2019-09-06 end of record is shown.



(a)

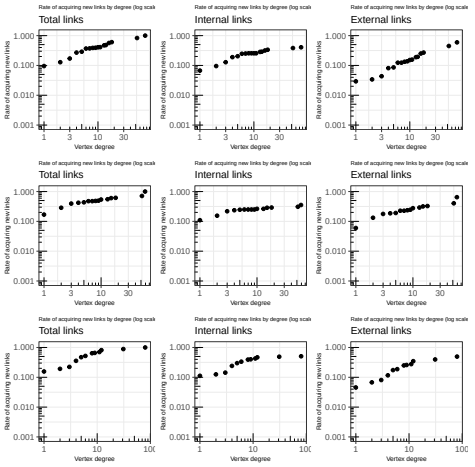


(b)

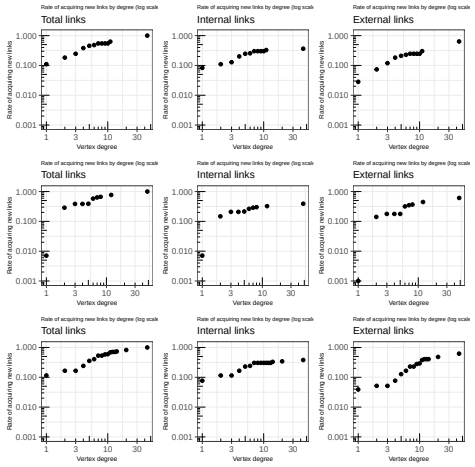


(c)

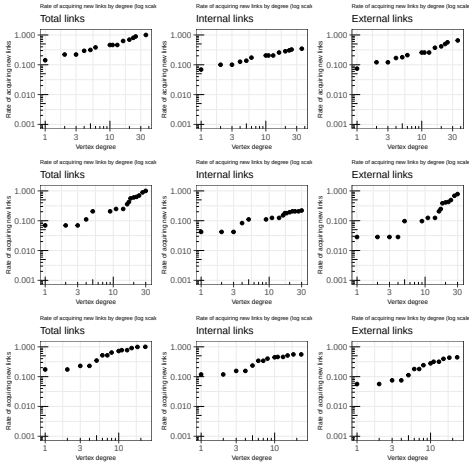
Figure A.15: The cumulative function for the three-month timeframe length of the commit-based data (edge-based split) for OpenSSL: In (a), visualizations for three three-month timeframes from 2002-02-23 until 2002-11-24 is shown. In (b), visualizations for three three-month timeframes from 2011-02-23 until 2011-11-24 is shown. In (c), visualizations for three three-month timeframes from 2019-05-26 until end of record is shown.



(a)

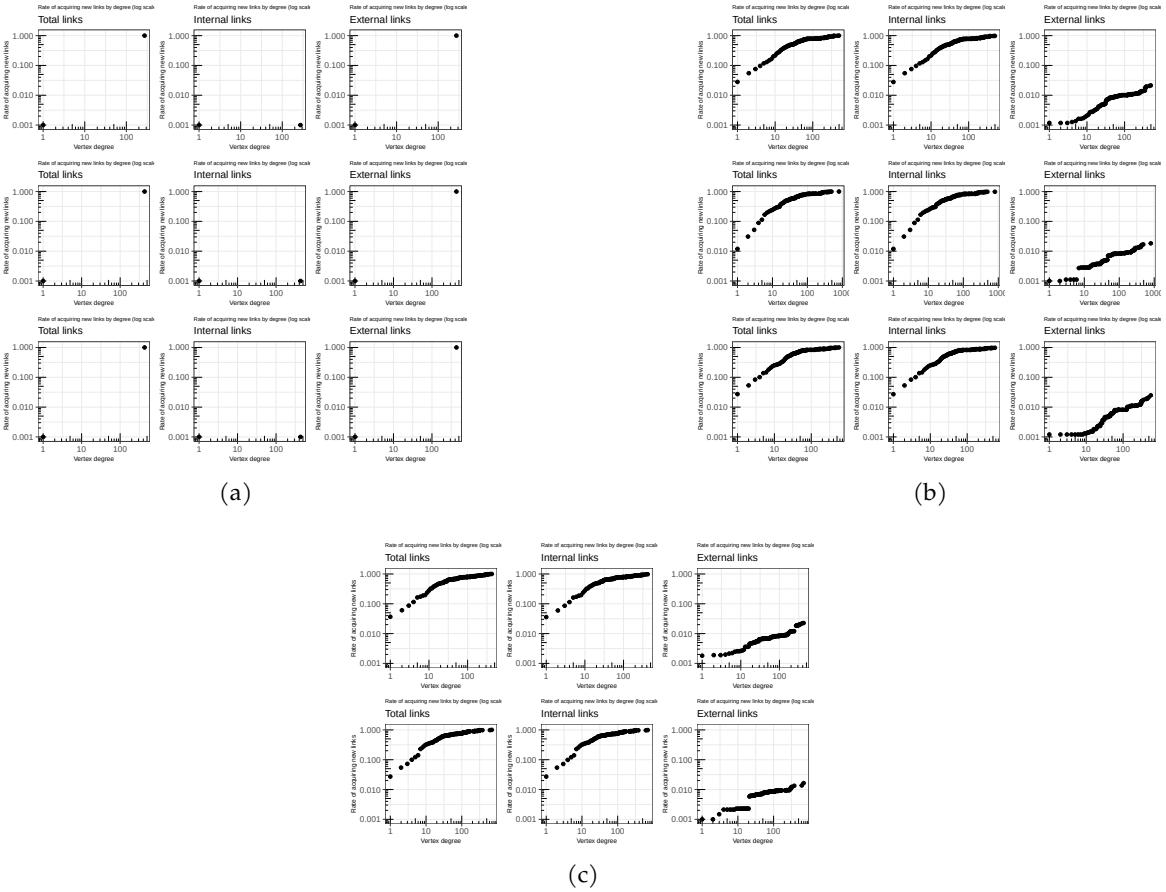


(b)



(c)

Figure A.16: The cumulative function for the three-month timeframe length of the mail-based data (edge-based split) for OpenSSL: In (a), visualizations for three three-month timeframes from 2002-06-13 until 2003-03-14 is shown. In (b), visualizations for three three-month timeframes from 2009-12-13 until 2010-09-13 is shown. In (c), visualizations for three three-month timeframes from 2017-06-13 until end of record is shown.

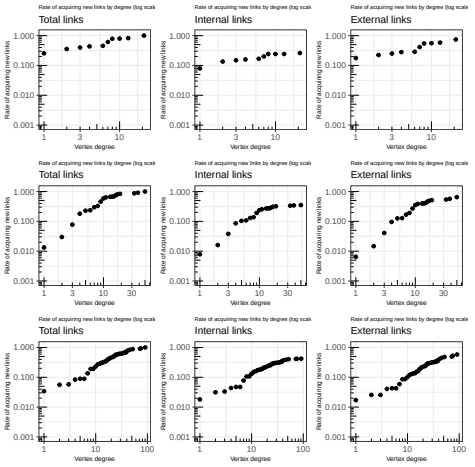


(a)

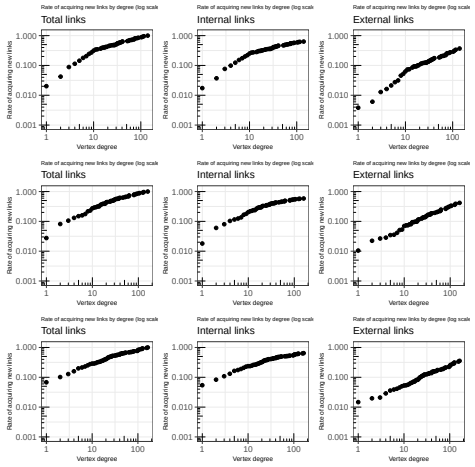
(b)

(c)

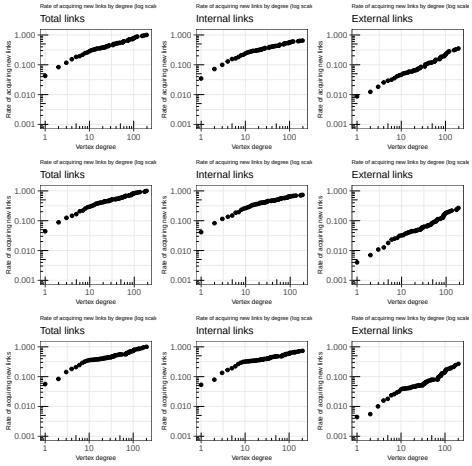
Figure A.17: The cumulative function for the three-month timeframe length of the commit-based data (edge-based split) for QEMU: In (a), visualizations for three three-month timeframes from 2003-05-21 until 2004-02-19 is shown. In (b), visualizations for three three-month timeframes from 2011-08-20 until 2012-05-20 is shown. In (c), visualizations for three three-month timeframes from 2019-11-19 until end of record is shown.



(a)

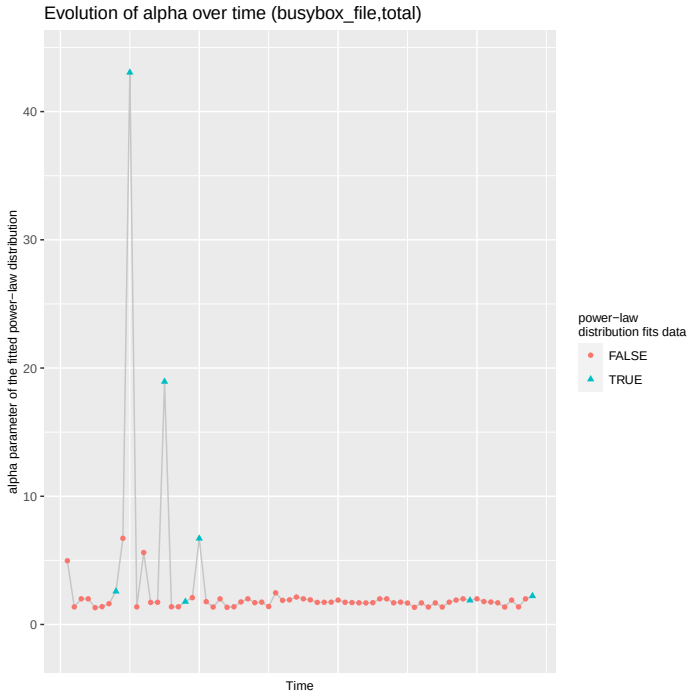


(b)

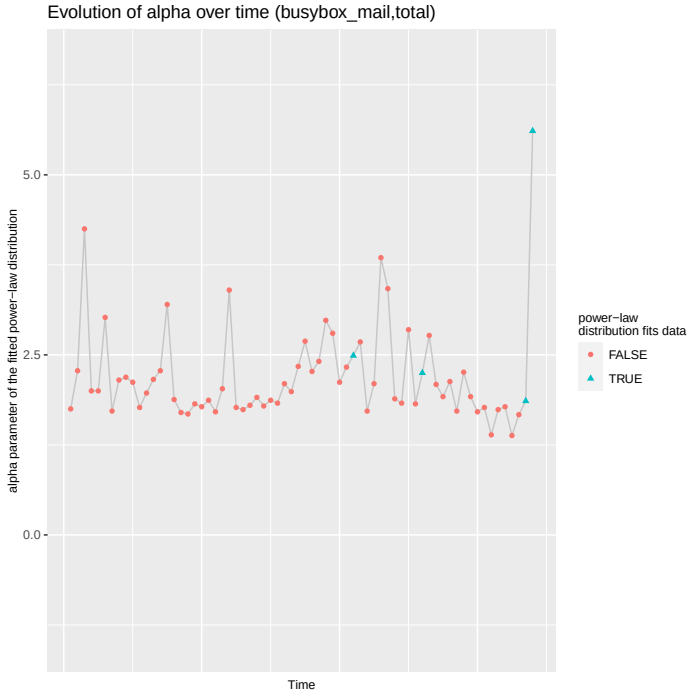


(c)

Figure A.18: The cumulative function for the three-month timeframe length of the mail-based data (edge-based split) for QEMU: In (a), visualizations for three three-month timeframes from 2003-12-19 until 2004-09-18 is shown. In (b), visualizations for three three-month timeframes from 2011-06-19 until 2012-03-19 is shown. In (c), visualizations for three three-month timeframes from 2019-09-19 until end of record is shown.

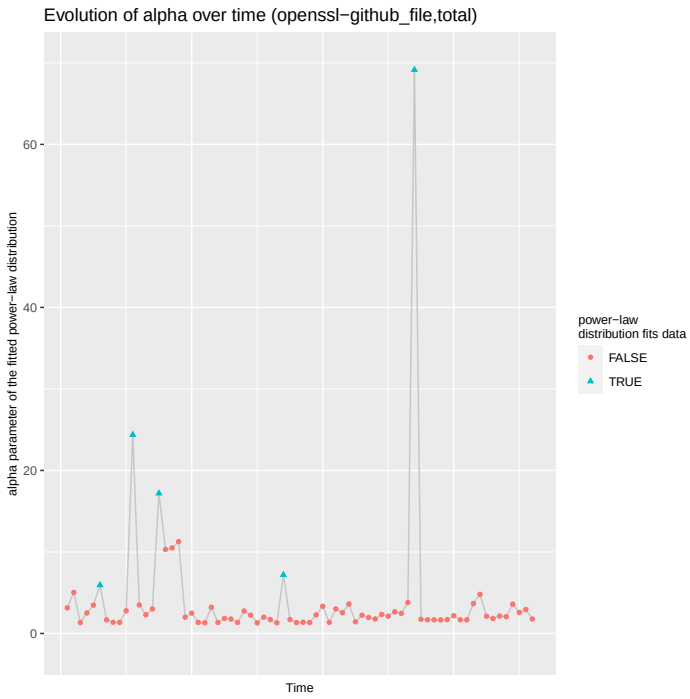


(a)

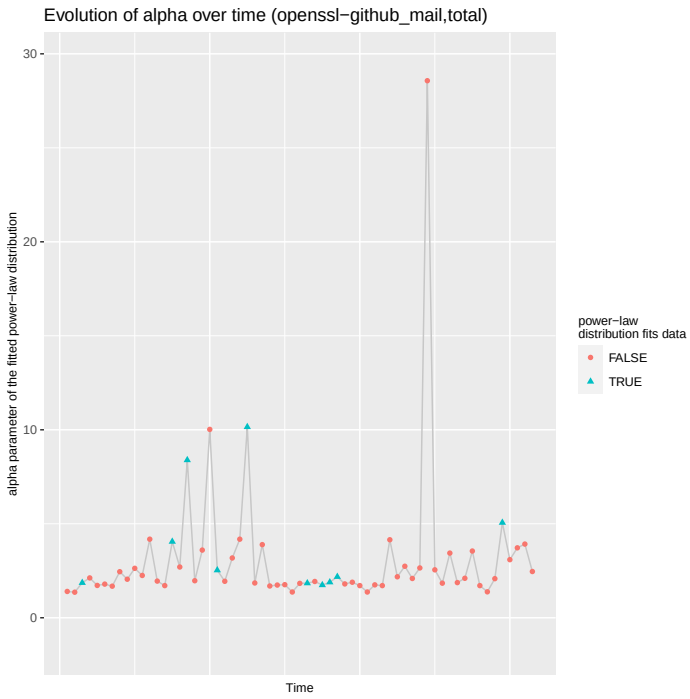


(b)

Figure A.19: The comparison between powerlaw-fit exponent α and preferential attachment for Busybox: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

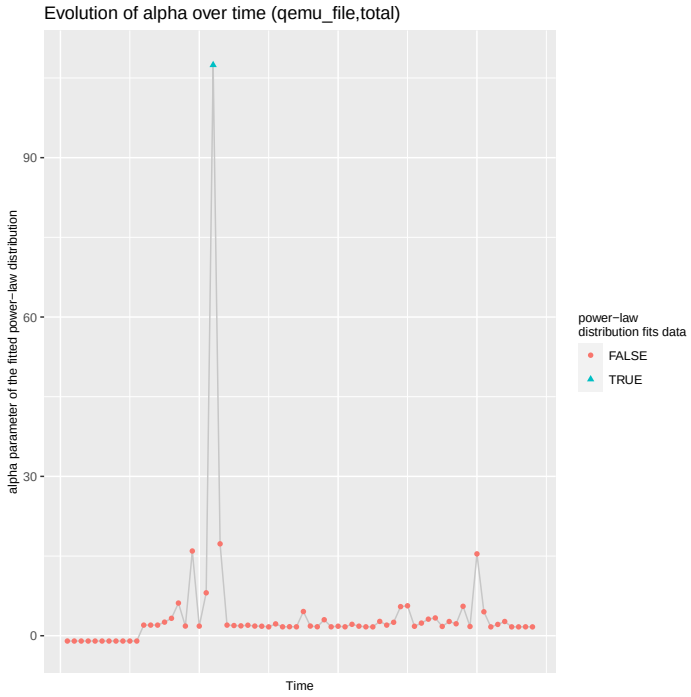


(a)

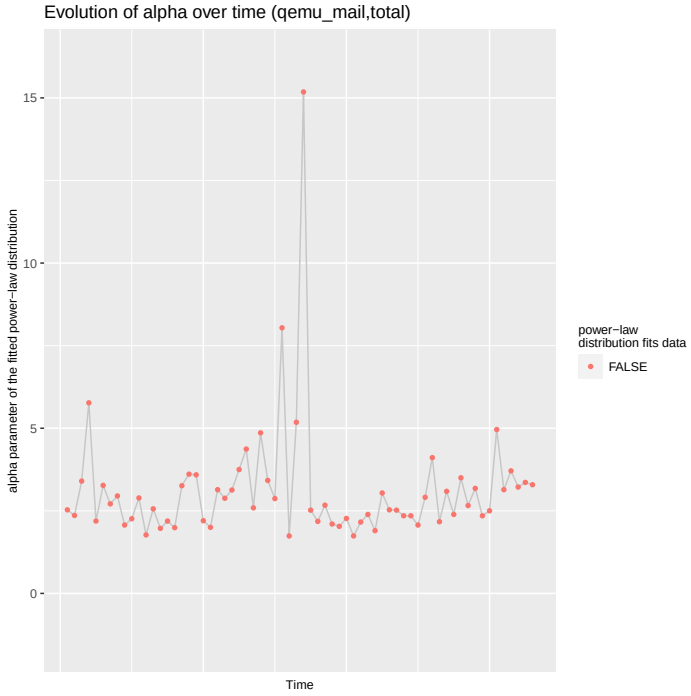


(b)

Figure A.20: The comparison between powerlaw-fit exponent α and preferential attachment for OpenSSL: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

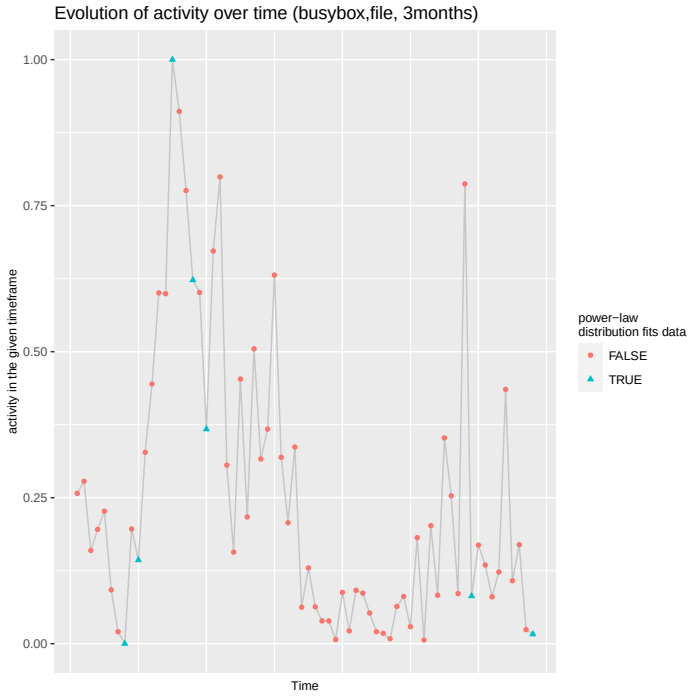


(a)

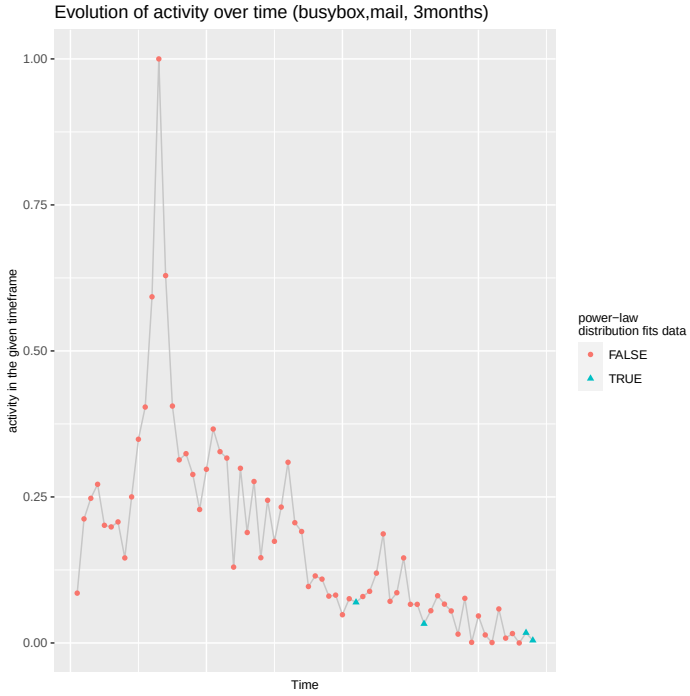


(b)

Figure A.21: The comparison between powerlaw-fit exponent α and preferential attachment for QEMU: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

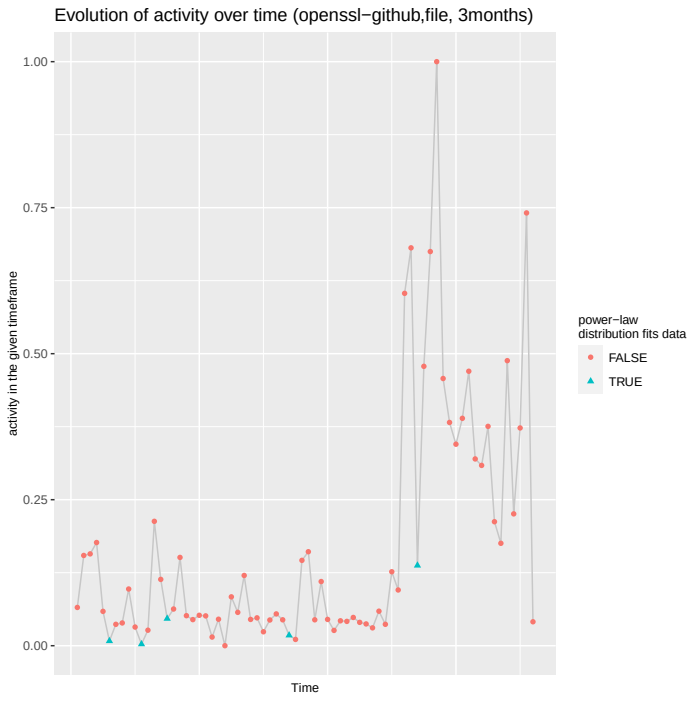


(a)

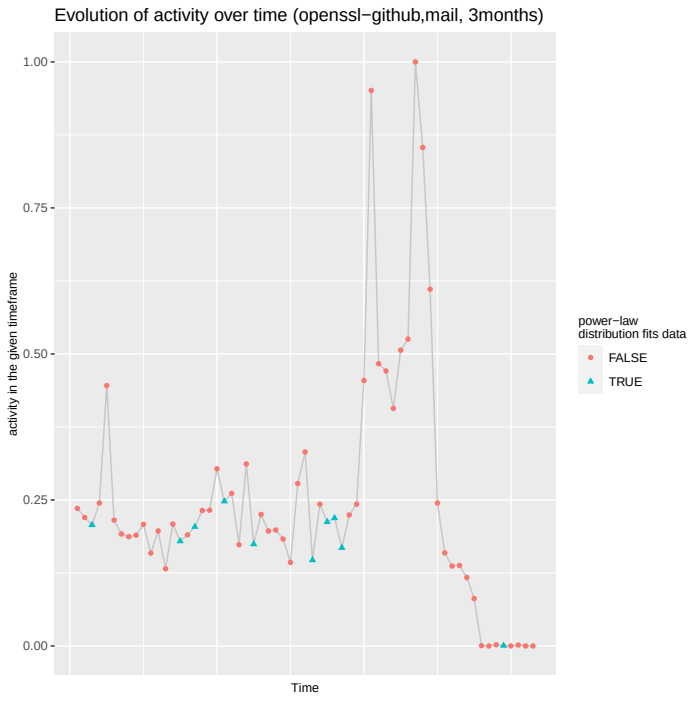


(b)

Figure A.22: The comparison between activity and preferential attachment for Busybox: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

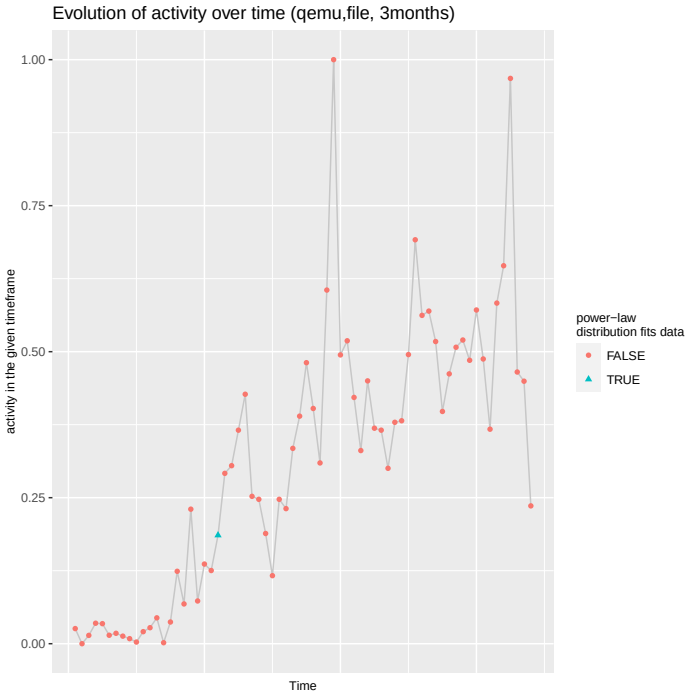


(a)

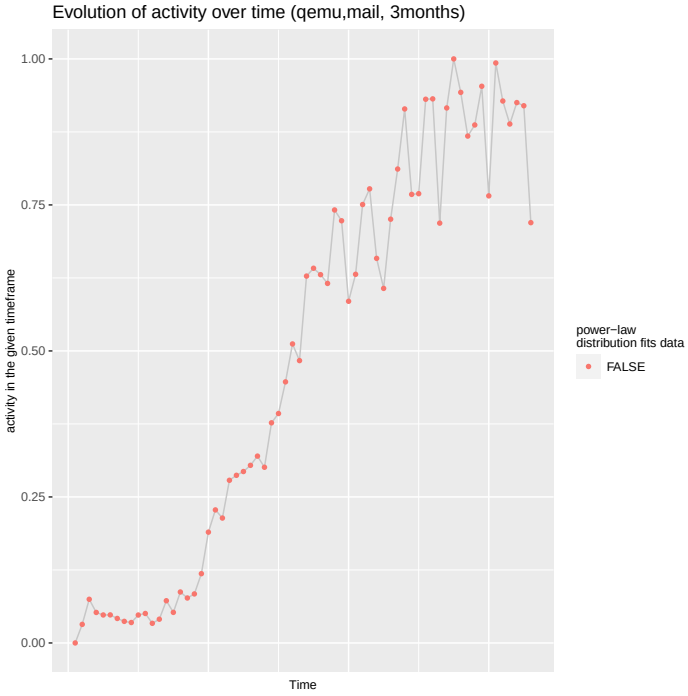


(b)

Figure A.23: The comparison between activity and preferential attachment for OpenSSL: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.



(a)



(b)

Figure A.24: The comparison between activity and preferential attachment for QEMU: In (a), the graph for the commit-based data for timeframes of a three-month length is shown. In (b), the graph for the mail-based data for timeframes of a three-month length is shown.

BIBLIOGRAPHY

[1] Morten Andersen-Gott, Gheorghita Ghinea, and Bendik Bygstad. "Why do commercial companies contribute to open source software?" In: *International Journal of Information Management* 32.2 (Apr. 2012), pp. 106–117. DOI: 10.1016/j.ijinfomgt.2011.10.003. URL: https://doi.org/10.1016/j.ijinfomgt.2011.10.003.

[2] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks." In: *Science* 286.5439 (Oct. 1999), pp. 509–512. DOI: 10.1126/science.286.5439.509. URL: https://doi.org/10.1126/science.286.5439.509.

[3] Albert-László Barabási. *Network Science*. Cambridge University Press, 2016. ISBN: 1107076269. URL: https://www.xarg.org/ref/a/1107076269/.

[4] Béla Bollobás. *Random Graphs*. Cambridge University Press, Aug. 2001. ISBN: 9780521797221.

[5] *Busybox*. 2012. URL: https://www.busybox.net/oldnews.html (visited on 06/14/2021).

[6] Guido Caldarelli. *Scale-Free Networks*. Oxford University Press, May 2007. ISBN: 9780199211517.

[7] A. Capocci, V. D. P. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli. "Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia." In: *Physical Review E* 74.3 (Sept. 2006). DOI: 10.1103/physreve.74.036116. URL: https://doi.org/10.1103/physreve.74.036116.

[8] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. "Power-Law Distributions in Empirical Data." In: *SIAM Review* 51.4 (Nov. 2009), pp. 661–703. DOI: 10.1137/070710111. URL: https://doi.org/10.1137/070710111.

[9] Kevin Crowston, Qing Li, Kangning Wei, U. Yeliz Eseryel, and James Howison. "Self-organization of teams for free/libre open source software development." In: *Information and Software Technology* 49.6 (June 2007), pp. 564–575. DOI: 10.1016/j.infsof.2007.02.004. URL: https://doi.org/10.1016/j.infsof.2007.02.004.

[10] Gábor Csárdi. *igraph: Network Analysis and Visualization*. R package version 1.2.4.1. 2019.

[11] Fabian Fagerholm, Alejandro Sanchez Guinea, Jay Borenstein, and Jurgen Munch. "Onboarding in Open Source Projects." In: *IEEE Software* 31.6 (Nov. 2014), pp. 54–61. DOI: 10.1109/ms.2014.107. URL: https://doi.org/10.1109/ms.2014.107.

[12] Mark D. Humphries and Kevin Gurney. "Network 'Small-World-Ness': A Quantitative Method for Determining Canonical Network Equivalence." In: *PLoS ONE* 3.4 (Apr. 2008). Ed. by Olaf Sporns, e0002051. DOI: 10.1371/journal.pone.0002051. URL: https://doi.org/10.1371/journal.pone.0002051.

[13] H Jeong, Z Néda, and A. L Barabási. "Measuring preferential attachment in evolving networks." In: *Europhysics Letters (EPL)* 61.4 (Feb. 2003), pp. 567–572. DOI: 10.1209/epl/i2003-00166-9. URL: https://doi.org/10.1209/epl/i2003-00166-9.

[14] Andrejs Jermakovics, Alberto Sillitti, and Giancarlo Succi. "Mining and visualizing developer networks from version control systems." In: *Proceeding of the 4th international workshop on Cooperative and human aspects of software engineering - CHASE '11*. ACM Press, 2011. DOI: 10.1145/1984642.1984647. URL: https://doi.org/10.1145/1984642.1984647.

[15] Mitchell Joblin, Sven Apel, and Wolfgang Mauerer. "Evolutionary trends of developer coordination: a network approach." In: *Empirical Software Engineering* 22.4 (Nov. 2016), pp. 2050–2094. DOI: 10.1007/s10664-016-9478-9. URL: https://doi.org/10.1007/s10664-016-9478-9.

[16] Tamas Nepusz and Gabor Csardi. *igraph: Network Analysis and Visualization*. R package version 1.2.4.1. 2020.

[17] Siobhán O'Mahony. "The governance of open source initiatives: what does it mean to be community managed?" In: *Journal of Management & Governance* 11.2 (June 2007), pp. 139–150. DOI: 10.1007/s10997-007-9024-7. URL: https://doi.org/10.1007/s10997-007-9024-7.

[18] Wonseok Oh and Sangyong Jeon. "Membership Herding and Network Stability in the Open Source Community: The Ising Perspective." In: *Management Science* 53.7 (July 2007), pp. 1086–1101. DOI: 10.1287/mnsc.1060.0623. URL: https://doi.org/10.1287/mnsc.1060.0623.

[19] *OpenSSL*. 2021. URL: https://www.openssl.org (visited on 06/14/2021).

[20] *QEMU*. 2020. URL: https://qemu-project.gitlab.io/qemu/ (visited on 06/14/2021).

[21] Maarten van Steen. *Graph Theory and Complex Networks*. Maarten van Steen, May 2021. ISBN: 9789081540612.

[22] Igor Steinmacher, Christoph Treude, and Marco Aurelio Gerosa. "Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects." In: *IEEE Software* 36.4 (July 2019), pp. 41–49. DOI: 10.1109/ms.2018.110162131. URL: https://doi.org/10.1109/ms.2018.110162131.

[23] Guowu Xie, Jianbo Chen, and Iulian Neamtiu. "Towards a better understanding of software evolution: An empirical study on open source software." In: *2009 IEEE International Conference on Software Maintenance*. IEEE, Sept. 2009. DOI: 10.1109/icsm.2009.5306356. URL: https://doi.org/10.1109/icsm.2009.5306356.