Physica A 413 (2014) 162–172

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Mechanism of asymmetric software structures: A complex network perspective from behaviors of new nodes

Lei Wang[a,b,*], Yu Wang[b], Yulong Zhao[b]

[a] State Key Laboratory of Software Development Environment, Beijing, China
[b] School of Computer Science and Engineering, Beihang University, Beijing, China

HIGHLIGHTS

- We found in- and out-degree distributions asymmetric in 223 Linux kernel modules.
- The preferential attachment behaviors are related with the age of nodes in Linux.
- The new nodes tend to cluster in complex software.
- We propose a novel model to generate simulated call graphs for Linux kernels.

ARTICLE INFO

Article history:
Received 30 September 2012
Received in revised form 4 May 2014
Available online 8 July 2014

Keywords:
Complex networks
Software evolution
Network model
Robustness

ABSTRACT

Studying the function call graphs in complex software can provide significant insights into the software evolution process. We found node in- and out-degree distributions asymmetric in the call graphs of 223 versions of Linux kernel modules (V1.1.0 to V2.4.35). Nodes newly introduced in these modules tended to attach to themselves (clustering) and existing high in-degree nodes. We proposed the $\alpha\beta$ Model to generate call graphs for different kernel modules. The model preserved asymmetry in the degree distributions and simulated the new node behaviors. Last, we discussed how the $\alpha\beta$ Model could be used effectively to study the robustness of complex networks.

1. Introduction

Many real-world networks have recently been found to be neither regular [1] nor random graphs [2], e.g., social or neural networks or the Internet, but complex networks [3]. These include, e.g., class diagrams [4,5], the collaboration graphs [6], the package dependency networks [7], the object graphs [8], the software component graphs [9] and the call graphs [10] in large-scale software. Most studies cover a range of software.

We, however, were interested in the evolution of software and selected the fs (file system), kernel, and mm (memory management) modules in the Linux kernel to study the changes between versions. After analyzing 223 Linux versions (from V1.1.0 to V2.4.35), we found that the in-degree distribution of the complex networks of the modules seemed to obey the power-law [11] while the out-degree distribution was similar to the power-law distribution with a cutoff [12]. The asymmetry between the in- and out-degree distributions seemed obvious.

Asymmetric structures lead to software fragility [13] and have a significant influence on the change propagation in software [14]. Myers points out that the asymmetry typically results from broad software reuse [6]. Ref. [4] argue that the

[*] Corresponding author at: School of Computer Science and Engineering, Beihang University, Beijing, China. Tel.: +86 10 82316284.
E-mail address: wanglei@buaa.edu.cn (L. Wang).

http://dx.doi.org/10.1016/j.physa.2014.06.062
0378-4371/© 2014 Elsevier B.V. All rights reserved.

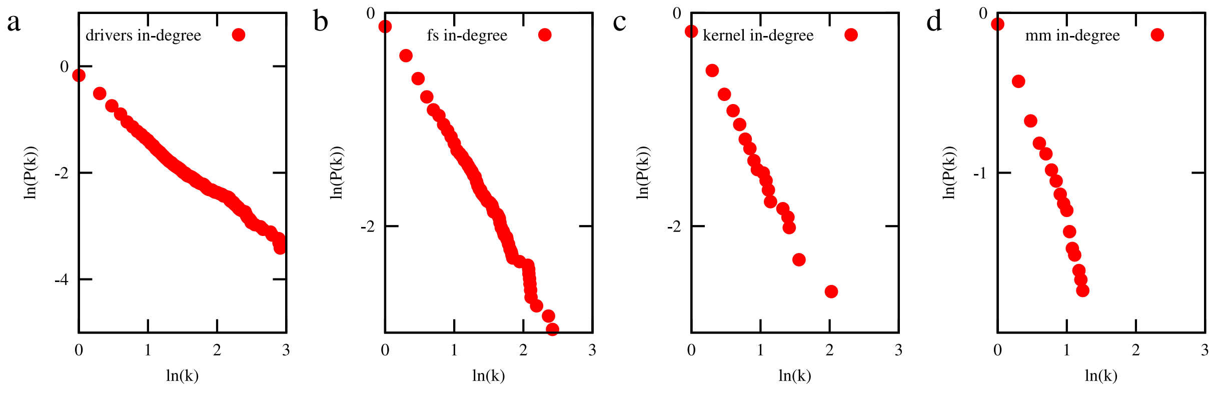

Fig. 1. Complementary cumulative in-degree distribution (Linux 2.2.6).

reasons for such asymmetry are rooted precisely in the economization of development effort and related costs. Ref. [15] note that the out-degree of the class is limited by the size of the class in the Object Oriented software systems, but the in-degree of the class does not have this limitation, which leads to the asymmetry. We found, however, that during the evolution of software the way new nodes are added could contribute to the asymmetry. In the four modules, new nodes exhibit high attachment tendency for in-degree but weak tendency for out-degree. We also found that new nodes are more likely to connect to new nodes (especially in the drivers module), which is different from the growing network model with copying (GNC) [16,17], where older nodes have greater priority to connect to the newly added nodes. We therefore believe that the behavior of the new nodes contributes to the asymmetry between in- and out-degree distributions. We further proposed the $\alpha\beta$ Model to generate simulated call graphs for Linux kernel modules, and found that the parameters of $\alpha\beta$ Model have a strong effect on the robustness of the network.

2. Call graphs of Linux kernel modules

The call graph describes the calling relationship among the functions of a program. A function, F, is represented by a node which has edges toward the nodes of the functions that F calls. The number of edges starting from F's node (the number of functions F calls) is the node's *out-degree* and that of those ending at the node (the number of functions that call F) its *in-degree*.

We generate a call graph for each kernel module and, by comparing the graphs of the same module of different versions, tried to understand how the Linux kernel evolves over time.

For a kernel module, each function defined in the module is a node. We added an edge from the caller to the callee when both functions are defined in the module. When the callee is defined in a different module, we added a node representing the callee's module and an edge from the caller to the node.

We collected call graph data by compiling Linux kernels using a modified version of GCC 3.4.6 [18] which produces call relationships for each C file. For clarity, we assigned sequence numbers in the chronological order to the 223 kernels between v1.1.0 and v2.4.35. i.e., the earliest kernel was assigned No. 1 and the latest No. 223.

3. Degree distribution

The complementary cumulative degree distribution (CCDD) of a graph, P_k, is defined as the probability that the degree of a node is greater than k.

$$P_k = \sum_{k'=k}^{\infty} p(k') \tag{1}$$

where $p(k')$ is the probability of a node's degree equals k'. We calculated the CCDDs for both the in- and out-degrees of the call graphs.

Fig. 1(a)–(d) show the in-degree CCDD for the drivers, fs, kernel, and mm modules of Linux 2.2.6, respectively. In each figure, the data seem to be well fit by a straight line. This strongly suggests that the in-degree CCDD follows a power-law and therefore the call graphs are scale-free complex networks.

A straight line indicates that P_k should be proportional to $k^{-(\gamma-1)}$ [19] where γ is a positive constant. Thus, we have:

$$P_k \sim k^{-(\gamma-1)}. \tag{2}$$

For complex networks, γ is called the degree exponent of a power-law distribution. It determines the slope of the straight line in the log–log plot.