Physica A 447 (2016) 520–526

Contents lists available at ScienceDirect

Physica A

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

Network of networks in Linux operating system

Haoqin Wang [a,b], Zhen Chen [c], Guanping Xiao [b], Zheng Zheng [a,b,*]

[a] State Key Laboratory of Software Development Environment, Beijing, 100191, PR China
[b] School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, PR China
[c] School of Electronic and Information Engineering, Beihang University, Beijing 100191, PR China

HIGHLIGHTS

- A network of networks about Linux operating system is constructed.
- Degree distribution similarity among the whole and component networks is observed.
- Manifestations in topology and function for the components have been observed.
- System failures make some nodes unreachable and visit new nodes at the same time.

ARTICLE INFO

Article history:
Received 13 September 2015
Received in revised form 26 October 2015
Available online 29 December 2015

Keywords:
Complex network
Linux operating system
Coupling relationship
System failure

ABSTRACT

Operating system represents one of the most complex man-made systems. In this paper, we analyze Linux Operating System (LOS) as a complex network via modeling functions as nodes and function calls as edges. It is found that for the LOS network and modularized components within it, the out-degree follows an exponential distribution and the in-degree follows a power-law distribution. For better understanding the underlying design principles of LOS, we explore the coupling correlations of components in LOS from aspects of topology and function. The result shows that the component for device drivers has a strong manifestation in topology while a weak manifestation in function. However, the component for process management shows the contrary phenomenon. Moreover, in an effort to investigate the impact of system failures on networks, we make a comparison between the networks traced from normal and failure status of LOS. This leads to a conclusion that the failure will change function calls which should be executed in normal status and introduce new function calls in the meanwhile.

1. Introduction

A wide range of real-world systems, such as the World Wide Web [1,2], scientific collaborations [3] and transportation infrastructures [4,5], can be described as complex networks where the entities are denoted by nodes and the relationships between entities are denoted by edges. In 1959, Erdös and Rényi introduced ER random network model [6], which has dominated the research on the complex network for decades. After that, the proposal of small-world models by Watts et al. [7] and scale-free properties by Barabási et al. [8] has attracted more and more researches on complex network in a wide field until now. In recent years, the researches on complex networks are flourishing, such as network modeling [9–11], epidemic spreading [12,13], cascading failures [14], traffic dynamics [15], evolutionary games [16–20], optimization process [21,22] and social dynamics [23–25]. One interesting direction is to study software systems [26–28] from the view of complex networks.

* Corresponding author.
E-mail address: zhengz@buaa.edu.cn (Z. Zheng).

http://dx.doi.org/10.1016/j.physa.2015.12.084
0378-4371/

Table 1
Basic properties of the six components. n is the number of nodes; m is the number of edges; $\langle k_{out}\rangle$ is the average out-degree; $\langle k_{in}\rangle$ is the average in-degree; $\langle f_{out}\rangle$ is the average external out-links and equals to the ratio of the number of external out-links to n; $\langle f_{in}\rangle$ is the average external in-links and equals to the ratio of the number of external in-links to n; C is the average clustering coefficient; E is the efficiency [41].

	arch	*drivers*	*fs*	*kernel*	*mm*	*net*
n	41 535	221 837	23 955	7656	3173	23 430
m	46 220	367 986	34 197	6494	3779	41 389
$\langle k_{out}\rangle$	1.112796	1.658813	1.427552	0.848224	1.190986	1.766496
$\langle k_{in}\rangle$	1.112796	1.658813	1.427552	0.848224	1.190986	1.766496
$\langle f_{out}\rangle$	2.276851	2.516708	4.250637	2.898511	3.453514	2.908451
$\langle f_{in}\rangle$	5.524088	0.532256	6.004258	13.74125	31.39994	6.820401
C	0.031023	0.043741	0.042347	0.029811	0.029721	0.035559
E	0.000495	0.000081	0.000733	0.001519	0.004204	0.000283

Software systems represent one of the most complex man-made systems and can be expressed as networks [29]. A wealth of study on software systems from the prospective of complex networks has been conducted over the past decade. Valverde et al. [30] presented the first evidence for the emergence of scaling and the presence of small world behavior in software systems. Myers [31] examined software systems as complex networks and provided a model of software evolution based on refactoring processes. Cai et al. [32] proposed a software mirror graph to record the dynamic information of software behaviors.

Operating system (OS) is one of the most important software systems and provides a basic executing environment for other software. Among various operating systems, Linux operating system (LOS), which is first released by Linus Torvalds in 1991, is well deployed in nearly all fields of our society nowadays. Understanding the inner structure of LOS is helpful for its maintenance and development. Additionally, the source open characteristic of LOS makes it convenient to analyze a software system as a complex network. In 2008, Zheng et al. [33] put forward two network growth models to better describe the properties of LOS. Recently, Gao et al. [34] analyzed the core components of LOS as a network and observed a scale-free phenomenon in it.

Furthermore, since LOS consists of several independent components, the interactions of which are essential for the execution of LOS. Thus, it is necessary and possible to explore the coupling relationships of components. The study of network of networks is a hot topic these days [35], and researchers had found that many realistic systems are interdependent on other systems. In 2010, Buldyrev et al. [36] developed a framework for analyzing the coupling relations of two interdependent networks. Whereafter, coupling relations have been extensively studied [37–39].

In this paper, we study the network of networks in LOS to explore the coupling relationships among components. The rest of this paper is organized as follows. Section 2 proposes the network modeling of LOS and the analysis of topological properties for six modularized components in LOS. Section 3 presents the coupling relationships of components both in topology and function. In Section 4, the impacts of system failures on LOS network are discussed. Finally, the conclusion of the work is given in Section 5.

2. Network modeling of LOS

LOS is mainly composed of six interacting components: *arch*, *drivers*, *fs*, *kernel*, *mm* and *net*. *arch* determines the feasible hardware that LOS can be installed in; *drivers* contains the device drivers; *fs* is the component of the disk and file system; *kernel* manages the processes in LOS; *mm* is the component of memory management and *net* serves networking [40]. We model LOS (Linux-3.16.1[1]) as a directed network, in which nodes represent functions and edges represent function calls. Fig. 1 illustrates a simple example for the network modeling.

It should be noted that the network of each component only contains the calls between functions of the same component. Additionally, the whole LOS network is constructed by component networks through the function calls between them.

Now we study the topological properties of the six components. It can be observed from Table 1 that the numbers of nodes for the components are quite different from each other. Among the six components, *drivers* is the largest one which is about 70 times larger than the smallest, i.e. *mm*. Each component has external links to communicate with other components. It shows that the value of $\langle f_{out}\rangle$ in *fs* is the largest, indicating that the realization of *fs* needs more interactions with other components. Moreover, the values of $\langle f_{in}\rangle$ in *kernel* and *mm* are larger, indicating that the functions in *kernel* and *mm* are called by other components more significantly. Another point worth mentioning is the efficiency. From the table, we can observe that the efficiency of *drivers* is extremely low. This is ascribed to two reasons. First, it is hard to find a loopback in the LOS network because there is no mutual function call between two functions in LOS. Besides, the number of nodes in *drivers* is the largest, which can account for the lowest efficiency in *drivers* dominantly. In the following, we will discuss the degree distributions of networks.

Fig. 2 exhibits the degree distribution of the whole system and the six components, which denotes the probability that the in-degree or out-degree of a randomly selected node is k. The figure indicates that the in-degree (or out-degree)

[1] We choose Linux-3.16.1 as our analyzing object, and it was the newest version when we started this research.