

SDN 控制平面功能模块化研究

侯文,陈佳,王洪超

(北京交通大学 电子信息工程学院 下一代互联网互联设备国家工程实验室,北京 100044)

摘要:随着网络规模的加大,网络节点不断增多,网络中需要进行管控和处理的数据也就越多,因此对控制平面的数据处理能力的要求也越来越高。如果控制平面只有一个集中的控制器来完成工作,其处理能力有限,在面对大规模复杂网络时,必将产生系统性能瓶颈。为了解决这一问题,提出了一种新的控制平面设计方案,即功能模块化控制平面。功能模块化控制平面将控制器内部按照功能划分成不同的子功能模块,各子功能模块能够独立部署,统一进行调度,对外实现逻辑集中化,内部又可以实现灵活扩展,进一步提高控制器的灵活性、可靠性以及整体性能。研究了功能模块化控制平面的总体模型设计和通信机制,最终通过搭建拓扑对功能模块化控制器进行测试,验证了该方案的可行性和有效性。

关键词:软件定义网络;控制平面;功能模块化;灵活部署

中图分类号:TP31

文献标识码:A

文章编号:1673-629X(2017)12-0023-05

doi:10.3969/j.issn.1673-629X.2017.12.006

Research on Modular and Functional SDN Control Plane

HOU Wen, CHEN Jia, WANG Hong-chao

(National Engineering Laboratory for NGI Interconnection Devices, School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing 100044, China)

Abstract: With the increase of the network scale and the increasing number of network nodes, the data to be controlled and processed in the network is increased, so the requirement for the data processing of the control plane is also higher and higher. If the control plane only has one single centralized controller to complete the work, its processing capacity is limited. When facing large-scale complex network, it will produce a systematic bottleneck. In order to solve this problem, a design scheme for new control plane, a functional and modularized control plane is proposed, which is divided into different sub function modules. Each one can be deployed independently and unified scheduling. The external logic can be centralized and the internal modules can be flexibly loaded. The ultimate goal is to improve the flexibility and reliability of the controller. The overall model design and communication mechanism of the modular control plane are studied. Finally, the functional modular controller is tested by constructing the topology and the feasibility and effectiveness of the proposed scheme is verified.

Key words: Software-Defined Network (SDN); control plane; functional modularity; flexible deployment

0 引言

随着网络的不断发展,传统网络的体系架构已经越来越难以满足对通信的“高速”、“高效”、“海量”的迫切需求,并且在可扩展性、安全性、移动性等方面暴露出严重不足。面临这一现状,对现有网络进行微小的改变已经不足以解决所有问题,因此需要一种全新的网络架构来综合解决当前网络中的众多弊端^[1]。2008年,斯坦福大学的 Nick McKeown 教授提出了 OpenFlow^[2],并逐渐推广到 SDN 概念。SDN 网络架

构^[3]将路由决策控制功能从原有传统网络中解耦出来,即实现网络中控制平面与数据平面的分离,由控制平面掌控网络的全局信息,实现对网络的控制,数据平面提供简单的数据转发功能。

在网络规模不庞大时,控制平面所需处理的请求量少,单一集中的控制器可以管理当前网络。单一控制器的代表有 NOX^[4]、Beacon^[5]等。然而当网络的规模不断加大,网络节点不断增多,发往控制器的流量也在不断加大,控制器所需处理和存储的数据量也就越

收稿日期:2016-11-24

修回日期:2017-03-27

网络出版时间:2017-08-01

基金项目:国家“973”重点基础研究发展计划项目(2013CB329100);国家自然科学基金重点项目(61232017)

作者简介:侯文(1991-),女,硕士研究生,研究方向为下一代互联网理论与技术;陈佳,博士,副教授,研究方向为下一代互联网理论与技术;王洪超,博士,讲师,研究方向为下一代互联网理论与技术。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20170801.1552.044.html>

来越大,而单一集中的控制器处理能力有限,部署不够灵活,扩展性差,无法管控越来越复杂的网络。文献[6-9]介绍了 SDN 控制器的发展现状,以及为了解决单一集中控制器所带来的众多问题,当前学术界提出的多种 SDN 控制平面的解决方案,例如采用多控制器架构。多控制器方案代表有如下几种:一种为去中心化结构,每个控制器的地位相等,如 HyperFlow^[10]、Onix^[11];一种为分层结构,将控制器分为不同等级,等级越高监控的范围越大,如 Kandoo^[12]。采用多控制器方案解决了单一控制器的性能瓶颈问题,但又带来了新的问题,其中主要包括控制器区域部署问题^[13]。例如对于给定的网络拓扑,如何进行分区,控制器该如何放置,每个控制器管理哪些交换机,等等。除了多控制器部署问题,多控制器方案面临的主要问题还涉及多个控制器之间该如何通信^[14],以及如何保持全局状态一致性等^[15]。

文中提出了一种新的控制器架构模型,即功能模块化的控制平面,用于解决单一集中的控制器所面临的问题。功能模块化的控制平面将原控制器中的不同功能抽象化,形成单独的功能模块,各功能模块能在不同的网络组件上进行灵活的部署和扩展,进一步提高控制器功能部署的灵活性和可扩展性。

1 控制平面功能模块化整体设计

为了提高控制平面部署的灵活性和可扩展性,功能模块化控制平面架构将原控制器中的不同功能模块进行抽象化处理,形成单独的功能模块,并且部署在不同的网络组件上。功能模块化的控制平面主要由以下几部分组成,其中包括一个控制器主控单元模块,多个子功能模块,如网络拓扑信息模块、路由计算模块、转发组件状态信息模块、用户信息模块和服务信息模块。

主控单元模块主要负责对各个功能模块的管理和控制,并且主控单元模块是控制平面和转发平面之间的接口。主控单元模块能够检测到有哪些功能模块完成注册,是否可以正常工作,并能够协调各子功能模块之间的工作。同时主控单元模块能够负责接收来自转发平面的消息,并根据消息类型,将不同的消息分配给相应的子功能模块进行处理。其他各个子功能模块可以部署在不同的组件上,完成各自的功能,并且各功能模块拥有自己的计算和存储资源,实现了控制平面的解耦合。

根据不同需求,实现同一功能的模块根据需要可能有多,由主控单元模块完成任务的分配,共同完成某一任务,减轻单一子功能模块的处理压力,使部署更灵活,并且提升了控制平面的整体性能。最终实现控制器主控单元与各子功能模块之间的协同工作,完成

对整个网络的管控,解决原有单一集中控制器的性能瓶颈和扩展性差的问题。控制平面的功能模块化设计如图 1 所示。

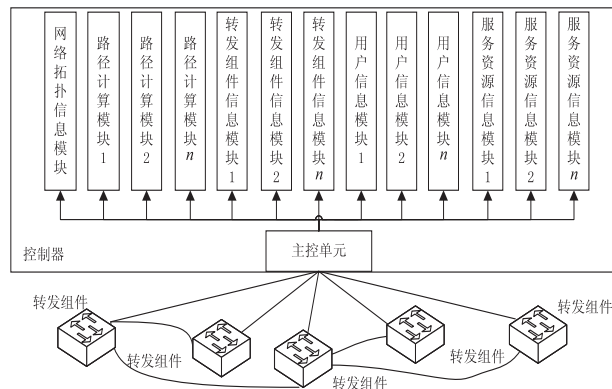


图 1 控制平面总体模型设计

2 控制平面各功能模块间的通信机制

2.1 控制器内部架构模型

控制器内部主要由主控单元模块、网络拓扑信息单元模块、路径计算单元模块、转发组件状态信息单元模块、用户信息模块和服务资源信息模块组成。控制器主控单元主要负责与各个功能模块间的信息交互,协调各功能模块的工作,以及与转发组件之间的通信。控制器主控单元是控制器内部的核心模块,也是控制器与外部的通信接口。网络拓扑信息模块负责实时保存网络中的拓扑信息,当网络中有服务请求时,路由计算模块会根据拓扑信息进行路径计算,完成资源的适配过程。转发组件状态信息模块负责存储各个转发组件的性能状态信息。用户信息模块负责存储网络中注册用户的信息,包括用户 ID、用户名称及用户权限等。服务资源模块负责存储网络中服务提供者可以提供的服务资源,包括服务 ID、域名、服务所需带宽、权限、类型等。

控制器的主控单元模块可以抽象为如下模型。主控单元由内部通信接口、外部通信接口、消息处理三部分组成。内部通信接口负责控制器内部主控单元与各功能模块的消息交互。外部通信接口负责与转发组件之间的消息交互。消息处理部分会根据内部通信接口与外部通信接口收到消息的类型进行相应处理,是主控单元模块的核心部分。

每个子功能模块可以抽象成如下模型。各子功能模块均由通信接口、消息处理及处理结果反馈几部分构成。

由通信接口负责接收与发送消息,消息处理部分负责根据不同的消息类型对消息进行相应的处理,如果需要发送消息反馈,则产生相应的反馈消息,由通信接口发送出去。

2.2 控制器内部通信流程

控制器内部通信涉及的消息类型包括功能模块注册消息 fun_hello, 信息查询与反馈消息 feature, 链路状态检测消息 echo。消息包格式的设计采用 TLV 形式, 用以区分消息类型和统一消息的格式。TLV: Type 类型, Length 长度, Value 值。类型部分采用主类型和子类型。

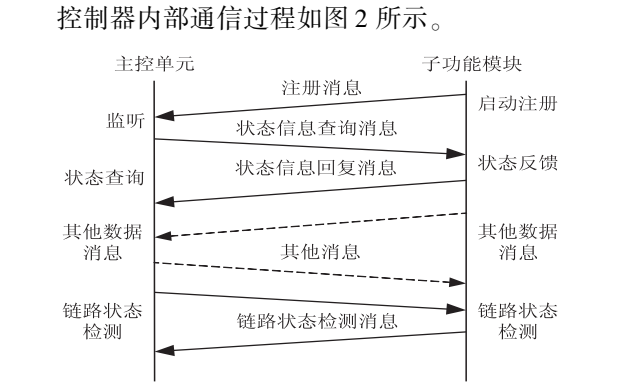


图 2 控制器内部通信流程

主控单元模块首先启动, 负责监听其他功能模块的连接。当有其他子功能模块发起连接请求时, 主控单元接受连接, 与子功能模块建立通信通道, 完成信息的交互。子功能模块与主控单元模块建立连接后, 子功能模块主动向主控单元模块进行注册, 实现功能模块的灵活加载。子功能模块发送注册消息 hello_fun 给主控单元模块, hello_fun 携带功能标识 fid, 包长度。主控单元模块可根据 hello_fun 消息中的功能标识来统计各个类型的功能模块分别有多少个, 并根据链路是否通畅来动态维护这一信息, 用于实现网络中有请求时任务的调度和分配。

控制器主控单元模块收到 hello_fun 消息后, 会向子功能模块发送信息查询消息, 即 feature_request 消息, 用于查询子功能模块的一些基本信息。子功能模块收到 feature_request 消息后, 向主控单元模块回应 feature_reply 消息, 用于报告自己的基本信息。feature_reply 消息中包含的内容有: 功能模块标识 fun_id、模块名称 fun_name、ip 地址、mac 地址、port 信息、工作状态信息等。

当主控单元模块与子功能模块间的连接建立、注册等一系列初步工作完成后, 控制器整体处于可以工作的待命状态。当网络中转发组件有消息发送给控制器时, 控制器会根据消息类型进行相应的处理, 实现主控单元模块和各子功能模块间的协同工作。当控制器主控单元与子功能模块的连接通道内没有消息发送时, 主控单元将向子功能模块定时发送 echo 请求消息, 子功能模块收到后给主控单元发送 echo 回复消息, 用于检测二者之间的通信是否通畅。如果在规定

时间和 echo 请求次数内, 主控单元模块没有收到子功能模块的 echo 回复消息, 则认为通信链路中断, 并将此类功能模块的个数减一。

2.3 控制器收到不同消息的处理过程

控制器主控单元模块负责接收由转发组件转发来的消息, 对接收的消息进行判断, 根据消息类型发送给相应的子功能模块, 由各子功能模块进行相应的处理。当主控单元收到转发组件的状态信息时, 将其存入转发组件状态信息模块。当主控单元模块收到用户的注册、注销或更新消息时, 将其发送给用户信息模块进行处理。当主控单元模块收到服务注册、服务注销或服务更新消息时, 将其交给服务资源信息模块进行相应的处理。当主控单元模块收到转发组件报告的拓扑信息时, 将其存入网络拓扑信息模块。当主控单元模块收到服务请求消息时, 首先在服务信息模块中查询是否有这一服务, 如果没有则返回该服务未提供, 服务请求失败; 如果存在这一服务, 服务信息模块向主控单元模块返回服务查询成功消息。主控单元模块向拓扑信息模块进行查询, 将拓扑信息和服务请求消息发送给路径计算模块, 由路由计算模块为该请求计算出一条合理可达的路径, 最终由主控单元模块下发给转发组件。

在各子功能模块注册阶段和链路探测阶段, 控制器主控单元模块动态感知每类功能模块的个数, 用于实现后续任务的分配。在适当的情况下, 通过对多个实现同种功能的模块的部署, 以提高控制平面的整体处理能力, 同时也增加了控制平面部署的灵活性。图 3 以控制器内部有两个路径计算功能模块为例, 展示了控制器收到网络中的服务请求消息时的工作过程。

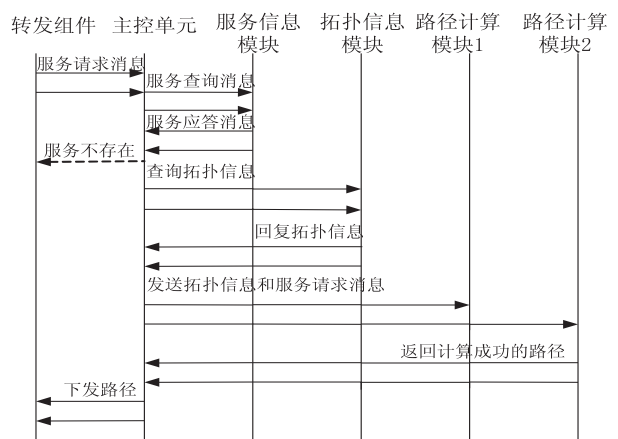


图 3 控制器内部处理流程

当网络中有多个服务请求时, 转发组件把服务请求信息发送给控制器。控制器主控单元模块接收到服务请求信息, 向服务信息模块进行服务信息查询, 若该服务不存在, 返回服务查询失败消息; 若存在该服务, 则服务资源信息模块返回服务查询成功消息。

当服务查询成功时,主控单元模块查询拓扑信息,并将拓扑信息和服务请求消息发送给路径计算模块。由于控制器内部有两个路径计算模块,主控单元模块将多个服务请求消息分别发送给路径计算模块 1 和 2。两个路径计算模块同时工作,进行路径的计算,并将计算结果返回给主控单元模块,由主控单元模块将路径下发给转发组件。若网络中存在到达该服务的路径,转发组件将此请求转发到对应的服务提供者,并将相应的服务返回给请求的主机。

3 实验测试

3.1 测试环境

测试过程在图 4 所示的拓扑中完成。测试环境包括一台控制器,三个转发组件,一台主机和一台服务器。

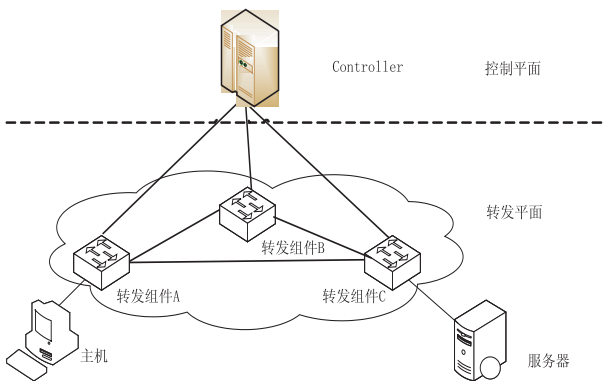


图 4 测试拓扑图

首先完成对控制器内部通信流程的功能性测试,其次对控制器内部结构的两种不同情况进行对比测试。第一种情况:控制器内部由一个主控单元模块、一个拓扑信息模块、一个路径计算模块、一个用户信息模块和一个服务信息模块组成。第二种情况:控制器由一个主控单元模块、一个拓扑信息模块、两个路径计算模块、一个用户信息模块和一个服务信息模块组成。

3.2 测试结果

针对图 4 所示的拓扑环境,通过 wireshark 抓包分析结果。抓包结果表明,控制器内部通信过程如下:控制器主控单元模块首先启动,各子功能模块向主控单元注册,发送注册消息即 hello_fun。主控单元模块收到注册消息后,向相应的功能模块发送状态信息查询消息 feature_request,子功能模块收到查询消息后,向主控单元模块回复 feature_reply,报告自己的相关信息。通过这种机制,实现控制器对其子功能模块的灵活加载。

当测试拓扑中有多个网络请求时,转发平面的交换机将多个服务请求消息发送给控制器的主控单元模块。主控单元模块进行服务查询得到结果服务存在,

然后查询拓扑信息。保持每秒发送的服务请求消息数量相同,对两种情况的控制器进行对比测试。当控制器内部只有一个路径计算模块时,主控单元模块将全部服务请求信息和拓扑信息发送给路径计算模块进行计算。当控制器内部有两个路径计算模块时,主控单元模块将服务请求消息近似平均分配给这两个子功能模块,子功能模块完成计算后将路径返回,并由主控单元模块将路径下发给转发组件。对比测试的结果如图 5 所示。

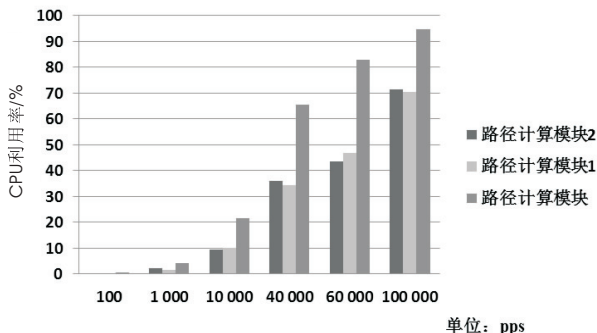


图 5 不同控制器的测试结果

从图 5 可以看出,当控制器内部只有一个路径计算模块时,该模块要全部完成对服务请求消息的路径计算,面临的压力也相对较大。当控制器内部有两个路径计算模块时,主控单元模块将服务请求消息进行分流,由两个路径计算模块分别进行计算。每个路径计算模块上承担的压力更小,能够更好地完成任务。

图 6 表明,当网络中发送服务请求消息速率相同时,两种情况下控制器路径下发的时延是存在差异的。当控制器中只有一个路径计算模块时,其路径下发时延要略大于控制器中有两个路径计算模块的下发时延。以上结果表明,当控制器中有多个相同功能的模块时,主控单元模块将任务进行分散,每个子功能模块的处理压力会相对减小,控制器的整体性能将会提高。

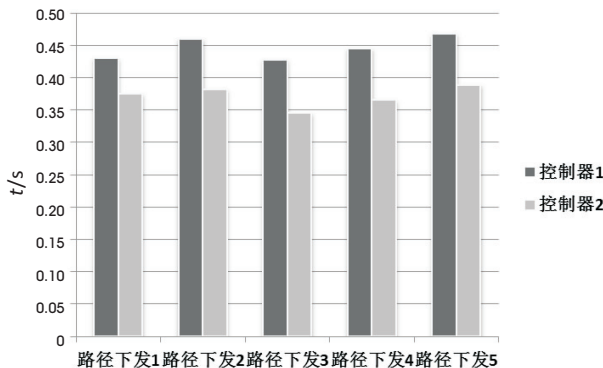


图 6 不同控制器的路径下发时延

4 结束语

分析了现有互联网面临的问题,以及目前 SDN 网

络架构中控制平面所面临的压力。对学术界目前提出的 SDN 控制器解决方案的优缺点做了简要介绍。同时提出了一种 SDN 控制器的优化解决方案,即功能模块化的控制平面架构。阐明了 SDN 控制平面功能模块化的设计思路,其中包括功能模块化控制平面的整体架构,控制器内部各部分组成,控制器内部通信流程,以及采用这种架构下的控制器在接收不同消息时的处理流程。最后通过搭建拓扑环境对控制器进行测试。结果表明,在这种架构下的控制平面能够对网络进行有效的管控,同时功能模块的部署也更灵活,提高了控制器的可扩展性。

参考文献:

[1] Feamster N, Rexford J, Zegura E. The road to SDN; an intellectual history of programmable networks[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 87-98.

[2] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[3] 张顺森, 邹复民. 软件定义网络研究综述[J]. 计算机应用研究, 2013, 30(8): 2246-2251.

[4] Gude N, Koponen T, Pettit J, et al. NOX: towards an operating system for networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105-110.

[5] 江国龙. SDN 控制器: Beacon 核心技术分析[J]. 程序员, 2014(2): 107-111.

[6] 李立龙, 吕光宏, 董永彬. 基于 OpenFlow 的 SDN 控制平面可扩展性综述[J]. 电子科技, 2015, 28(1): 171-175.

[7] 房秉毅, 张歌, 张云勇, 等. 开源 SDN 控制器发展现状研究[J]. 邮电设计技术, 2014(7): 29-36.

[8] Dixit A, Hao F, Mukherjee S, et al. Towards an elastic distributed SDN controller[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 7-12.

[9] Jimenez Y, Cervello-Pastor C, Garcia A J. On the controller placement for designing a distributed SDN control layer[C]//Networking conference. [s. l.]: IEEE, 2014: 1-9.

[10] Tootoonchian A, Ganjali Y. HyperFlow: a distributed control plane for OpenFlow[C]//Internet network management conference on research on enterprise networking. [s. l.]: USENIX Association, 2010: 3.

[11] Koponen T, Casado M, Gude N, et al. Onix: a distributed control platform for large-scale production networks[C]//USENIX symposium on operating systems design and implementation. Vancouver, BC, Canada: USENIX, 2010: 351-364.

[12] Yeganeh S H, Ganjali Y. Kandoo: a framework for efficient and scalable offloading of control applications[C]//Workshop on hot topics in software defined networks. [s. l.]: ACM, 2012: 19-24.

[13] 鲜永菊, 朱佳, 鲁昭男. 基于 SDN 的多控制器部署策略的研究[J]. 电视技术, 2016, 40(6): 78-84.

[14] Lin P, Bi J, Chen Z, et al. WE-bridge: west-east bridge for SDN inter-domain network peering[C]//IEEE conference on computer communications workshops. [s. l.]: IEEE, 2014: 111-112.

[15] 李军飞, 兰巨龙, 胡宇翔, 等. SDN 多控制器一致性的量化研究[J]. 通信学报, 2016, 37(6): 86-93.

(上接第 22 页)

on quality software. [s. l.]: IEEE, 2012.

[4] TAO W, LI W H. Naive Bayes software defect prediction model[C]//International conference on computational intelligence and software engineering. Wuhan, China: IEEE, 2010: 1-4.

[5] ELISH K, ELISH M. Predicting defect-prone software modules using support vector machines[J]. Journal of Systems and Software, 2008, 81(5): 649-660.

[6] SHEPPERD M, BOWES D, HALL T. Researcher bias: the use of machine learning in software defect prediction[J]. IEEE Transactions on Software Engineering, 2014, 40(6): 603-616.

[7] QUAH T S, THWIN M M T. Application of neural networks for software quality prediction using object-oriented metrics[C]//International conference on software maintenance. [s. l.]: IEEE, 2003: 589-590.

[8] ZHENG J. Cost-sensitive boosting neural networks for software defect prediction[J]. Expert Systems with Applications, 2010, 37(6): 4537-4543.

[9] FENTON N E, NEIL M. Software metrics: roadmap[C]//Pro-

ceedings of conference on the future of software engineering. [s. l.]: [s. n.], 2000: 357-370.

[10] 刘旸. 基于机器学习的软件缺陷预测研究[J]. 计算机工程与应用, 2006, 42(28): 49-53.

[11] BELKIN M, NIYOGI P. Laplacian eigenmaps for dimensionality reduction and data representation[J]. Neural Computation, 2003, 15(6): 1373-1396.

[12] BRUCHER M, HEINRICH C, HEITZ F. A metric multidimensional scaling-based nonlinear manifold learning approach for unsupervised data reduction[J]. Eurasip Journal on Advances in Signal Processing, 2008(1): 1-12.

[13] LU H, CUKIC B, CULP M. A semi-supervised approach to software defect prediction[C]//Computer software and applications conference. [s. l.]: IEEE, 2014: 416-425.

[14] MENZIES T, GREENWALD J, FRANK A. Data mining static code attributes to learn defect predictors[J]. IEEE Transactions on Software Engineering, 2007, 33(1): 2-13.

[15] 陈家强. 软件缺陷预测中数据预处理技术研究[D]. 南京: 南京大学, 2014.