

# 基于网站访问行为的匿名爬虫检测

邹建鑫,李红灵

(云南大学 信息学院 计算机科学与工程系,云南 昆明 650000)

**摘要:**通过分析和研究网络爬虫访问网页内容的行为,针对恶意网络爬虫伪装成浏览器访问网站难以甄别、网站日志检测工具不支持匿名网络爬虫检测等问题,总结了一些基于机器人排斥协议和基于爬虫行为的恶意网络爬虫检测算法。通过这些网络爬虫检测算法的启发,提出一种基于爬虫行为的检测匿名爬虫算法。该算法主要根据人为访问网站与网络爬虫访问网站时间的长短、访问的周期等,对网络爬虫进行检测,同时对算法进行了实验验证。实验数据来自一个服务器的网络日志。应用 Python 对实验数据进行处理,从而对网络匿名爬虫进行检测,并与当前主流的匿名网络爬虫检测算法进行比较。结果表明,该算法能够检测出并发量小的匿名的网络爬虫。

**关键词:**网络爬虫;网络机器人排斥协议;网站访问行为;匿名爬虫检测

中图分类号:TP393.08

文献标识码:A

文章编号:1673-629X(2017)12-0103-05

doi:10.3969/j.issn.1673-629X.2017.12.023

## Anonymous Crawler Detection Based on Web Access

ZOU Jian-xin, LI Hong-ling

(Department of Computer Science and Engineering, School of Information Science and Engineering,  
Yunnan University, Kunming 650000, China)

**Abstract:** By analysis and study of web crawler accessing web page, some detection algorithms of malicious web crawler are summarized based on robot exclusion protocol and crawling, aiming to the problem that it is difficult to identify website accessing from malicious web crawler disguised as a browser, and that web log detection tools don't support anonymous web crawler detection. In consideration of above algorithms, a new one to identify the camouflage web crawler is proposed based on crawling. It detects the web crawler mainly according to the length of access time and access cycle of website accessing form both human and crawler, and is verified by an experiment, the data of which is from a server web log. The experimental data are processed by Python for anonymous crawler detection. Compared with mainstream detection algorithm of anonymous web crawler, the proposed algorithm can detect the small amount of concurrent anonymous web crawler.

**Key words:** web crawler; robot exclusion protocol; website access; camouflage crawler detection

## 0 引言

为了批量抓取互联网网站中的特定链接和内容,20 世纪 90 年代出现了一种自身拥有搜索策略的网络程序,将其命名为网络爬虫(Web Crawler)<sup>[1]</sup>。网络爬虫首先广泛应用于搜索引擎(如 Google 的 Googlebot),其主要任务是从互联网抓取内容,将抓取的内容经过整理后放入自己公司搜索引擎的索引库中,以供用户进行内容查询。

随着大数据<sup>[2]</sup>时代的到来和数据挖掘<sup>[3]</sup>的应用,网络爬虫也随之迅猛发展,从而使其成为全球互联网的一大公害。除了一些专门从事搜索引擎<sup>[4]</sup>的公司,

如:Google、微软、百度外,国内大多数大型门户网站都开发了自己的搜索引擎,如:搜狐、腾讯、网易等等,此外还有很多国内外不知名的搜索引擎,甚至一些学生组织的科研机构等完成的小型搜索引擎。网络上充斥着如此数量巨大的网络爬虫<sup>[5]</sup>,会影响网站安全<sup>[6]</sup>,阻塞网络<sup>[7]</sup>。

现今主流的网络爬虫检测<sup>[8]</sup>方法主要为三种。第一:限制单个 IP/api token 的访问量,比如 15 分钟限制访问 180 次;第二:蜜罐资源,在网站页面中嵌入浏览器不能看到的资源,如果有 IP 访问了就视为爬虫;第三:利用网站日志分析工具 awstats 等对网络日志进行

收稿日期:2016-12-08

修回日期:2017-04-10

网络出版时间:2017-08-01

基金项目:国家自然科学基金资助项目(61562090)

作者简介:邹建鑫(1991-),男,硕士,研究方向为计算机网络和信息安全;李红灵,副教授,研究方向为计算机网络和信息安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170801.1557.076.html>

定期分析,找出并发量大的匿名访问 IP。但这几种方法对于并发量小的,只是抓取特定内容的匿名爬虫显得无能为力,只有从其访问行为上才能区别出来。

## 1 网络爬虫概述

### 1.1 网络爬虫的爬取方式

网络爬虫实质上是一种驻留在某个主机之内的软件<sup>[9-11]</sup>,并不会在互联网中移动。主体思想是通过主机(如:个人电脑、服务器等)发送 HTTP 请求来获取网页,应用自身的搜索策略从获取的网页中得到超链接(URL),又通过超链接不断地递归下去。其行为类似于人使用浏览器访问英特网中的网页,但其行为是根据爬取策略来自动完成和实现的。

爬虫的一般递归策略为:首先选定要爬取的地址,将这些地址放入列表当中;其次根据列表中的地址自动发送 HTTP 请求,获取相应的网页之后,再对网页进行分析,得到更多的 URL;最后对得到的 URL 进行分析并过滤掉列表中已有的 URL,将没有的 URL 增加到自己的爬取列表中。重复以上步骤,递归地进行爬取操作。

### 1.2 网络机器人排斥标准

为了使网络爬虫和 Web 服务器协同工作,在保障网站安全运行的前提下,允许网络爬虫抓取网站上的某些数据,1994 年众多网络爬虫作者以及网络爬虫爱好者共同商讨,提出了网络机器人排斥标准。

网络机器人排斥协议<sup>[12]</sup>(Standard for Robot Exclusion, SRE):由网站管理员或者网站设计人员在服务器的根目录下,设置 robots.txt 文件来实现。在这个文件中明确标明了网络爬虫不可以抓取本网站的资源信息,文件的内容由一条或者多条记录联合构成。每一条记录包含一个或者多个 User-Agent,每个 User-Agent 后包含一个或者多个 Disallow 行。User-Agent 所记录的是一些网络爬虫的名称,表示这些爬虫可以访问该网站中的网页。Disallow 的值是一些禁止访问本网站的网页链接名称,在同一个记录的 User-Agent 行中列出的网络爬虫将不能访问这些网页链接名称。

网络机器人排斥协议的 Robots.txt 范例如图 1 所示。

User-agent:
Disallow:

图 1 Robots.txt 范例

## 2 遵循网络机器人排斥标准的网络爬虫检测

Web 服务器日志记录<sup>[13]</sup>是用户访问该站点发出请求所产生的数据,对于遵守机器人排斥协议标准的

网络爬虫可以直接通过访问记录将其检测出来,方法有以下 5 种。

### 2.1 Robots.txt 检测

网络机器人排斥协议中明确规定,爬虫访问站点的网页前,必须先访问 Robots.txt 文件。也就是首先要有一个请求资源 robots.txt,从服务器管理的角度,在管理 Web 服务器时,如果一个网络爬虫没有请求 robots.txt 就直接按照自身的搜索策略来对网站进行爬取,那么就视这个网络爬虫为恶意爬虫。

### 2.2 User-Agent 检测

对于每个遵守机器人排斥协议的网络爬虫,在发送网页请求时,一定会将自己的 User-Agent 封装在请求头当中,而且自身的 User-Agent 是网络当中特有的,不会是某个浏览器的名称。由此可以通过 User-Agent 来检测是否为爬虫程序。但是一般恶意爬虫会通过设置 User-Agent 的内容来伪装浏览器。

### 2.3 IP 地址检测

IP 地址记录了访问网站站点的源地址,根据源地址是否大量发送链接请求,可以用来鉴别是否为网络爬虫,确定为网络爬虫之后可以通过封禁 IP 地址来达到控制网站访问的目的。但目前大部分爬虫程序 IP 地址是动态变化的,或者是直接通过代理 IP 对网站进行爬取,所以 IP 地址检测的漏检率与误判率比较高。

### 2.4 Referer 字段检测

一些简易的爬虫程序不会在发送的 Request 请求当中封装 Referer 字段,Web 服务器可以通过检查 HTTP 请求头中是否具有 Referer 字段来识别网络爬虫。在地址栏中直接输入链接,产生的 Request 请求中没有这个字段,这个字段也可以被封装进 Header 当中,所以这种方法的检错率也可能较低。

### 2.5 Method 域检测

部分网络爬虫为了减少请求服务器的开销回应,在 Method 域中采用 HEAD 方法来请求。如果在网络日志当中有某个 IP 拥有很多的 HEAD 请求,则此网站访问者可能被判定为网络爬虫。

### 2.6 总结

以上 5 种方法主要针对遵循网络机器人排斥标准的静态的网络爬虫的检测,但各有其优缺点,结合几种方法同时考虑会提供准确度。但是网络机器人排斥协议所定义的一些内容并不能够限制恶意网络爬虫对网站的抓取,因为恶意网络爬虫现今基本上都封装成假冒的 Request 请求,尽最大努力地模仿浏览器<sup>[14]</sup>的行为,甚至出现了一些专用工具。如 Phantomjs,这是一个没有界面的浏览器,专门解析网站中动态生成的 JS 文件以及 AJAX 文件,Phantomjs 本身就是浏览器,通过它的访问都可以伪装成一个真正用户的请求。只有通

过服务器日志的行为分析才能够区分出这是一个伪装爬虫。检测出来后,服务器管理员可以通过 IP 封锁<sup>[15]</sup>等方式对这些恶意爬虫进行相应处理。

3 伪装成浏览器的网络爬虫检测

遵守网络机器人排斥协议的网络爬虫很容易通过网站日志等进行判断和区分。而现今许多网络爬虫是通过伪装成浏览器来进行,这就不能够简单地从网站日志当中将其区分出来,需要根据其行为进行动态分析来进行甄别。

3.1 设置 robots.txt 陷阱

在服务器根目录的 robots.txt 文件代码中,加入不允许访问的网页<sup>[16]</sup>,例如/admin/hidden.html。这个网页是不存在或者是无关紧要的文件,而且这个网页或者文件只在 robots.txt 里面被提及,一个网络爬虫只有通过 robots.txt 才能获知这个网页。非恶意爬虫遵循 robots.txt 的规定,不会去访问这个网页或者文件,也不会知道这个网页或者文件的 URL,访问该网页或者文件的网络爬虫极可能就是那些视图利用 robots.txt 抓取网站机密文件的恶意爬虫。

3.2 蜜罐技术

网络爬虫通常缺乏足够的能力去辨别网页代码中各链接的真伪<sup>[17]</sup>。根据这个策略,可以在网页代码中加入一些在浏览器中不能显示的或是显示不明显的,但是可能被网络爬虫访问的隐藏链接,通过这种方式来捕获恶意爬虫。例如:链接的文本采用与 background 一样的颜色,在浏览器中很难看出这个链接,正常的网站访问一般情况是不会访问到这个链接的。如果爬虫爬取了这个链接,即被视为恶意爬虫。另外,如果只是检测那些不遵守网络爬虫限制策略的爬虫,可以在 robots.txt 中将这些链接进行 Disallow 描述。

3.3 基于行为模式检测

3.3.1 伪装成浏览器的爬虫行为模式

对于未知的、匿名的、恶意的网络爬虫,有几个共性:不遵守相关的协议、伪装自己的 User-Agent、构造虚假的 Referer 信息、设置代理 Proxy IP。为了识别它们,不能依靠已经存在的协议,协议都可能被假冒,只有从其访问模式上进行识别。

从访问模式上看,网络爬虫与人为通过浏览器访问有很大的区别,网络爬虫有自己早已设定好的访问策略,而人为浏览是漫无目的的。例如:网络爬虫的搜索策略有广度优先和深度优先,并且可以设置访问深度,人为浏览网站一般跨度不可能如此之大。

人为通过浏览器访问某个网站,步骤是:在浏览器当中输入要访问网站的 URL,浏览器解析 URL,然后向解析后的 IP 地址的服务器发送链接请求;Web 服

务器收到请求后,检查服务器上是否存在该资源,如果存在,就将资源传送,如果没有则返回错误信息,如 404 等信息;通过网络传输之后,浏览器收到了网站服务器发送的文件,接着就会对所收到的文件进行解析。接收到的文件是单一文件体,则直接在浏览器页面当中显示,例如动画。接收到的文件是 ASPX、JS 或者是 HTML 文件,分析出这些文件当中需要的嵌入对象,包括:JS 脚本、CSS、图片、声音等,按照嵌入对象顺序,发送嵌入对象请求;服务器收到这些请求后,将对象发送;浏览器收到嵌入对象后,将这些嵌入对象装进 ASPX、JS、HTML 文件中,通过页面处理,将一个完整的网页呈现出来。综上所述,人为通过浏览器访问某个网页,在服务器的访问日志中,拥有多条请求记录。

网络爬虫的访问模式则与此有很大的不同:从等待爬取的 URL 列表当中取出一个 URL,假定所请求的是一个 HTML,接下来把一些必要的参数封装进链接请求,将封装好的请求发送出去;服务器收到请求后,将 HTML 框架文件发送回去;爬虫在收到文件之后,对接收到的文件进行解析,将该 HTML 中的超链接进行分析,需要的就增加到自己的访问队列中,而对于其他的一些链接和嵌入对象,根据网络爬虫功能、处理方法的不同,选择直接丢弃或是选择特定的对象进行抓取。但是有一个明显的特点就是并不会马上向服务器发送对象请求,所以网络爬虫的访问,一次请求只在服务器日志中留下一条记录信息。

爬虫访问网站和浏览器访问网站的行为差异如图 2 所示。

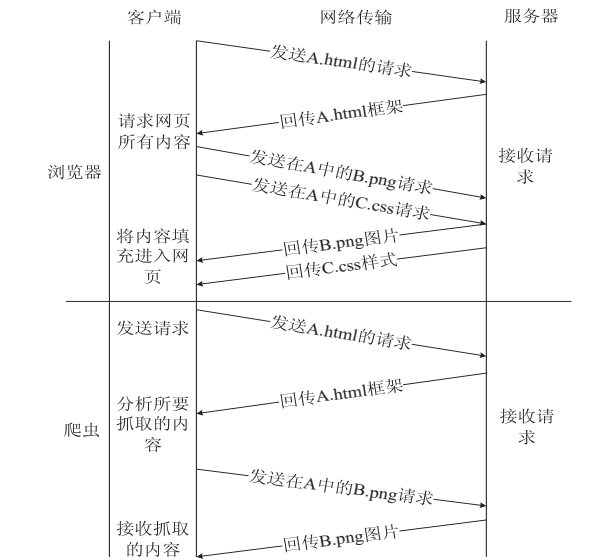


图 2 爬虫访问网站和浏览器访问网站的行为差异

3.3.2 基于爬虫访问行为模式算法

网络上一般网页的访问都遵循先请求网页框架,然后再向服务器发送请求网页中的内容。所以一个网页由主体的 html<sup>[18]</sup> 文件以及嵌入式的网页成员构成。



定义 1 将页面的成员定义为： $W = \langle \text{weblink}, \{\text{webmember1}, \text{webmember2}, \dots, \text{webmembern} \} \rangle$

在二元组当中,weblink 为互联网访问的一般 URL,webmember 为页面中所有嵌入式文件对象的 URL 集合。两者的不同是服务器访问日志记录中访问状态不同,如果说请求到了 weblink 页面,服务器日志记录中所标识的请求状态为 200,而请求到 webmember 对象,其服务器日志记录中的请求状态为 304。200 表示请求成功,而 304 表示请求重定向到本网站当中的资源。

其中,嵌入式的 webmember 包括:FRAME 之类的框架网页文件;在 HTML 中的众多单一文件,例如图片、声音、动画等;JAVASCRIPT 之类语言的脚本文件;CSS 之类的样式文件;Applet Class 之类的 CODE 文件。例如,该算法实验数据网页 index.aspx 中包括一个 CSS 页面/all.css 存在服务器的根目录当中,还有另外九个图片文件存在根目录下的 images 目录下。根据定义构造出 index.aspx 文件的二元组为:

$W = \langle \text{index.aspx} \{ \text{all.css}, \text{head.jpg}, \text{index\_r2\_c1.jpg}, \text{index\_r3\_c6.jpg}, \text{index\_r2\_c22.jpg}, \text{spacer.gif}, 44\_r1\_c1.jpg}, 44\_r2\_c2.jpg}, 44\_r2\_c4.jpg}, 44\_r11\_c1.jpg \}, 10 \rangle$

定义 2 将每个访问成员访问过的资源定义为:  
 $U = \langle \text{uid}, \text{IP}, \text{User-Agent}, \{ \text{request1}, \text{request2}, \text{request3}, \text{request4} \}, n \rangle$

其中,uid 为访问者的 ID;IP 为访问者的 IP 地址;User-Agent 为用户代理;request 为在一段时间内向服务器发出的请求; $n$  为请求定义 1 的二元组的个数。

定义 3 对每一条网站访问记录定义为:  
 $R = \langle \text{IP}, \text{Request}, \text{Time}, \text{User-Agent}, \text{Status} \rangle$   
其中,IP 为日志访问记录中的访问者 IP;User-Agent 为用户代理;Time 为访问时间;Status 为返回状态码。

算法 1:伪装爬虫检测算法伪代码。  
Procedure BuildUser;  
List  $R$  (定义 3)  $W$  (定义 1)  
UserList  $U$  (定义 2) user(定义 2)  
Begin  
temp $\leftarrow 0$  #设置临时变量作为访问点  
For  $i = \text{temp}; i < R.\text{length}();$  #访问每一条日志  
temp $\leftarrow \text{temp} + 1$  #设置临时变量作为访问点  
If  $R[i].\text{request} == W.\text{url}$  then  
#一条日志当中的请求为待检测的页面  
user $\leftarrow \text{new User}()$  #新建一个访问用户  
user.request.append( $R[i].\text{request}$ )  
#将当前请求元素放入用户请求表中  
For  $j = i + 1; R[i].\text{ip} == R[j].\text{ip} \& \& R[i].\text{ag} == R[j].\text{ag}; j++$   
万方数据

访问下一条日志判断条件为 ip、agent 相同  
If  $R[j].\text{time} - R[i].\text{time} < 30$  then  
#两者记录时间小于 30 s  
user.request.append( $R[j].\text{request}$ )  
#将当前日志请求放入请求表  
End If  
temp $\leftarrow j$  #临时变量记录访问到哪一个元素  
End For  
U.append(user) #将访问用户记录到用户表当中  
End If  
End For  
For each  $u \in U$  do #访问用户表中的每一个元素  
If  $u.\text{request}$  在  $W$  中的 request 数为 0 then  
#如果用户请求中没有检测页面的请求元素,则认定此用户为爬虫  
Result.append( $u$ ) #将用户放入结果列表  
End If  
End For  
Return result #返回检测出的爬虫列表  
End

通常情况下,人为访问某个网站<sup>[19]</sup>,通过浏览器来访问几乎是同时对页面元素进行请求,时间一般在 0~5 s 以内,一般不超过 30 s。如果是网络爬虫,请求到页面之后请求页面内容通常会超过 30 s。最后得出的结果包含一个用户表,而有用的信息就是访问 URL 成员的数量,如果数量为 0,几乎可以判定为伪装的网络爬虫。

4 实验结果及验证

对一个网站一段时间的日志进行验证,实验结果如表 1 所示。

表 1 算法检测出来的爬虫

IP 地址	User-Agent
172.16.54.38	Mozilla/5.0 (compatible;+MSIE+6.0;+Windows+NT+5.1)
218.18.228.88	Mozilla/5.0 (compatible;+MSIE+5.0;+Windows+98;+DigExt)
10.125.98.193	Mozilla/5.0 (compatible;bingbot/2.0;+http://www.bing.com/bingbot.htm)
172.16.20.136	Mozilla/5.0 (Windows;+U;+Windows+NT+5.1;+zh-CN;+rv:1.7.5)+Gecko/20041124+Firefox/1.0
61.135.145.206	Baiduspider+(+http://www.baidu.com/search/spider.htm)
172.16.31.66	Mozilla/5.0 (compatible;+MSIE+6.0;+Windows+NT+5.1;+MyIE2)
172.16.19.80	Mozilla/5.0 (compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+Maxthon)
211.66.188.147	Mozilla/5.0 (compatible;+MSIE+6.0;+Windows+98)

首先用网站分析工具 SEO 分析这一段时间内的网络日志,所得结果主要有页面抓取和爬虫分析(仅针对爬虫的项),如表 2 所示。

表 2 网站日志分析工具得出的页面抓取结果

序号	页面	状态码	爬虫	抓取量
1	/info_service/tousu/index. asp	200	百度	6
2	/institution/tuanwei/html/shjwzh/shi-jianwenzhai16. html	404	百度	1
3	/special _ topic/harvest/gj01/pindex. htm	200	百度	1
4	/news/newshtml/schoolNews/20151221105623. asp	200	百度	
5	/info _ pub/zsjy/zsxx/question/login. asp	200	百度	1
6	/jxzl/rczp/rczp. htm	404	百度	1
7	/index. asp	200	百度	1
8	/news/newshtml/schoolNews/20151222111524. asp	200	百度	
9	index. asp	200	微软 bing	1

实验当中仅仅测试的是/index. asp 页面,可以看到状态码为 200,这是已经申明自己机构的爬虫,通过 awstats 或者其他的网络日志分析工具都可以查出,而匿名的爬虫这些分析工具并不能分析出来。综合网络日志分析工具,看实验结果中的 IP 请求次数,以及在链接当中是否跳转过页面而没有请求过本站的资源,如表 3 所示。

表 3 实验所得 IP 请求次数

IP 地址	链接次数	是否跳转页面
172. 16. 54. 38	9	是
218. 18. 228. 88	1	否
172. 16. 20. 136	12	是
172. 16. 31. 66	6	是
172. 16. 19. 80	22	是
211. 66. 188. 147	15	是

实验数据当中有四个 IP 是属于本地局域网,有两个外网,其中有一个是只发送过一次链接请求,这与微软必应爬虫、百度爬虫的行为方式相同,只对网站爬取过一次(此为检测方法的主要特征),初步可以判定此 IP 为匿名爬虫。然后在接下来一段时间的日志当中进行此 IP 的查找,在访问此网站的页面资源时只有一个请求,并没有其他的资源请求,而且此 IP 会周期性地访问网站一次,凡是正常的用户一次访问可能会出现只请求页面问题,长期反复的只请求页面而不请求资源,即可确定此 IP 为匿名的网络爬虫。

该方法的提出主要针对现今网络爬虫分析工具匿名爬虫检测功能的不足,在用 awstats 进行网站日志分

析时查找匿名网络爬虫。实验结果当中就用了国内 SEO 网站分析工具对网络日志进行分析,分析结果表 3 已经显示,对于开源的国外公认的 awstats 也是基本类似的功能。对此类分析工具所得出的结果只能检测出并发量高的网络爬虫,而不能检测出并发量低的网络爬虫。

该方法与主流的限制单个 ip/api token 的访问量进行比较,限制 IP 主要的工作是限制而不是检测,大众点评网站当中就用到了。如果少量地抓取大众点评上的信息,限制策略并没有用。

该方法与蜜罐资源即上文所提及的使用隐藏资源进行比较,不必在设计网站当中加入冗余的东西来增加网站的复杂性,而且对于只抓取特定内容的网络爬虫蜜罐资源并不能够检测出网络爬虫。

5 结束语

通过对爬虫规范的解析,以及爬虫行为的分析,检测匿名爬虫现今只能从爬虫并发量、爬虫访问网站的行为等方面入手。匿名爬虫并发量可以通过设置服务器手段完成,而从爬虫访问行为方面并没有太多的工具以及手段,所以完善现今的网络日志分析工具对匿名爬虫的功能,网站才能够得到有力的保护。

参考文献:

[1] Cho J,Garcia-Molina H. The evolution of the web and implications for an incremental crawler[C]//Proceedings of 26th international conference on very large data bases. San Francisco,CA, USA; Morgan Kaufmann Publishers Inc,2000:200-209.

[2] 王 珊,王会举,覃雄派,等. 架构大数据:挑战、现状与展望[J]. 计算机学报,2011,34(10):1741-1752.

[3] Witten I H, Frank E, Hall M A. Data mining:practical machine learning tools and techniques[M]. [ s. l. ];Elsevier Science,2011.

[4] Croft B, Metzler D, Strohman T. Search engines-information retrieval in practice[J]. Computer Journal,2011,54(5):831-832.

[5] Raghavan S, Garcia-Molina H. Crawling the hidden web [C]//Proceedings of 27th international conference on very large data bases. San Francisco,CA,USA;Morgan Kaufmann Publishers Inc,2001:129-138.

[6] Tan Pangning,Kumar V. Discovery of web robot sessions based on their navigational patterns[J]. Data Mining and Knowledge Discovery,2002,6(1):9-35.

[7] 郭伟刚,鞠时光. 电子商务网站中 Web Robot 的检测技术[J]. 计算机工程,2005,31(23):219-211.

[8] Najork M,Heydon A. High-performance web crawling[M]//

## 5 结束语

为解决多管理域下 SDN 网络中控制器发生单点故障的问题,就目前复杂多变的网络环境,提出了跨管理域的故障监控和恢复的解决方案。分析了控制器故障对 SDN 网络的重大影响和必要性,给出了故障恢复解决方案的应用场景。设计了 SDN 控制器故障的监测方法,然后基于故障监测的结果设计了多管理域下的 SDN 控制器故障恢复策略,通过备用控制器的选择和交换机迁移的手段实现了 SDN 控制器的故障恢复。最后通过实验测试了控制器故障恢复策略的正确性,证明该方案是可行的,具有较高的实用价值。

### 参考文献:

- [1] 沈苏彬. 软件定义联网的建模与分析[J]. 南京邮电大学学报:自然科学版,2014,34(3):1-9.
- [2] 黄 韬,刘 江. 软件定义网络核心原理与应用实践[M]. 北京:人民邮电出版社,2014.
- [3] 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.
- [4] BRAUN W, MENTH M. Software-defined networking using OpenFlow: protocols, applications and architectural design choices[J]. Future Internet, 2014, 6(2): 302-336.
- [5] WANG Feng, WANG Heyu, LEI Baohua, et al. A research on carrier-grade SDN controller[C]//International conference on cloud computing and big data. [s. l.]: IEEE, 2014: 168-174.
- [6] 林萍萍,毕 军,胡虹雨,等. 一种面向 SDN 域内控制平面可扩展性的机制[J]. 小型微型计算机系统, 2013, 34(9):

1969-1974.

- [7] YAZICI V, SUNAY O M, ERCAN A O. Contolling a software-defined network via distributed controllers[C]//Proceedings of the 2012 NEM summit. Turkey: [s. n.], 2012: 16-20.
- [8] TOOTOONCHIAN A, GANJALI Y. HyperFlow: a distributed control plane for OpenFlow[C]//Proceedings of the 2010 internet network management conference on research on enterprise networking. [s. l.]: USENIX Association, 2010, 3.
- [9] YEGANEHS H, GANJALI Y. Kandoo: a framework for efficient and scalable offloading of control application [C]//Workshop on hot topics in software defined networks. [s. l.]: ACM, 2012: 19-24.
- [10] 姚 龙. 软件定义网络控制器容量及部署问题研究[D]. 合肥:中国科学技术大学, 2015.
- [11] Open Network Fundation. Software-defined networking: the new norm for networks[EB/OL]. (2012-04-13)[2016-02-08]. <https://www.opennetworking.org>.
- [12] 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.
- [13] CHAN Y C, WANG Kuochen, HSU Y H. Fast controller failover for multi-domain software-defined networks [C]//European conference on networks and communications. [s. l.]: IEEE, 2014: 370-374.
- [14] OREBAUGH A, RAMIREZ G, BURKE J, et al. Wireshark & ethereal network protocol analyzer toolkit[J]. Jay Beales Open Source Security, 2007, 12(2): 523-540.
- [15] LANTZ B, O'CONNOR B. A mininet-based virtual testbed for distributed SDN development[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(5): 365-366.

(上接第 107 页)

- Handbook of massive data sets. [s. l.]: Kluwer Academic Publishers, 2001: 25-45.
- [9] 详解网络爬虫与 Web 安全[J]. 计算机与网络, 2012, 38(12): 38-39.
- [10] Shkapenyuk V, Suel T, Shkapenyuk V, et al. Design and implementation of a high-performance distributed web crawler [C]//Proceedings of 18th international conference on data engineering. [s. l.]: IEEE, 2002: 357-368.
- [11] Cho J, Garcia-Molina H, Page L. Efficient crawling through URL ordering [J]. Computer Networks & ISDN Systems, 1998, 30(1-7): 161-172.
- [12] 张 峰,付 俊,杨光华,等. Web 访问日志安全分析技术研究[J]. 北京邮电大学学报, 2014, 37(2): 93-98.
- [13] 李佳欣,潘 伟. PhantomJS 在 Web 自动化测试中的应用[J]. 计算机光盘软件与应用, 2013, 16(18): 76-77.
- [14] 梁雪松. 网络机器人对网络安全的影响及其应对策略[J].

信息安全与通信保密, 2008(8): 94-96.

- [15] Linnér L, Arborelius L, Nomikos G G, et al. Locus coeruleus neuronal activity and noradrenaline availability in the frontal cortex of rats chronically treated with imipramine: effect of alpha 2-adrenoceptor blockade [J]. Biological Psychiatry, 1999, 46(6): 766-774.
- [16] 范纯龙,袁 滨,余周华,等. 基于陷阱技术的网络爬虫检测[J]. 计算机应用, 2010, 30(7): 1782-1784.
- [17] Artail H, Masri Z A, Sraj M, et al. A dynamic honeypot design for intrusion detection [C]//IEEE/ACS international conference on pervasive services. [s. l.]: IEEE, 2004: 95-104.
- [18] 常红要,朱征宇,陈 烨,等. 基于 HTML 标记用途分析的网页正文提取技术[J]. 计算机工程与设计, 2010, 31(24): 5187-5191.
- [19] 贾梦青. 基于用户访问行为分析的网站分类研究[D]. 郑州: 郑州大学, 2009.