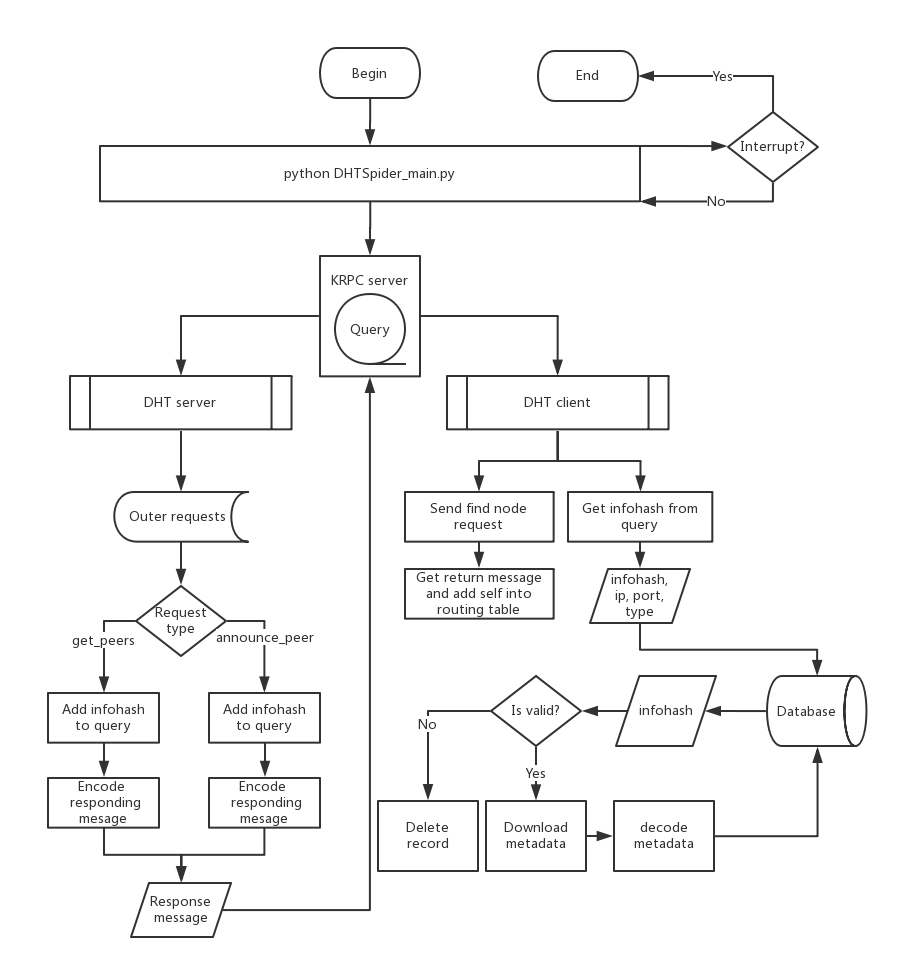
### DHT网络爬虫模块



**图3-2-4 DHT网络爬虫程序设计图**

要实现DHT协议的网络爬虫，主要分3步。第一步是得到资源信息(infohash，160bit，20字节，可以编码为40字节的十六进制字符串)，第二步是确认这些infohash是有效的，第三步是通过有效的infohash下载到BT的种子文件，从而得到对这个资源的完整描述。

其中第一步是其他节点用DHT协议中的get\_peers方法向爬虫发送请求得到的，第二步是其他节点用DHT协议中的announce\_peer向爬虫发送请求得到的，第三步可以有几种方式得到，比如可以去一些保存种子的网站根据infohash直接下载到，或者通过announce\_peer的节点来下载到，

KRPC服务器实现了DHT server 和 DHT client二者的功能，KRPC 服务器遵循KRPC 协议。KRPC 协议是由 bencode 编码组成的一个简单的 RPC 结构，他使用 UDP 报文发送。一个独立的请求包被发出去然后一个独立的包被回复。这个协议没有重发。它包含 3 种消息：请求，回复和错误。对DHT协议而言，这里有 4 种请求：ping，find\_node，get\_peers 和 announce\_peer。

DHT client有两个功能，第一个是向其他节点发出find\_node的请求，以加大DHT server在其他节点的留存几率。在一个爬虫中主要也是两个地方用到find\_node，第一是初始路由表时，第二是验证K桶是否存活时。第二个是通过获取到的infohash与宣称该infohash的ip建立连接，通过BitTorrent协议获取到该资源的metadata，主要包括：资源的名称，资源的md5值，资源的大小等信息。

DHT server通过接收其他节点发来的get\_peer请求，这个是向服务器请求是否有其他节点正在下载info\_hash对应的资源。在爬虫中有节点向自己请求时不仅像个正常节点一样做出回应，还需要以此资源的infohash为机会尽可能多的去认识更多的节点。另一个请求为announce\_peer，这是向一个节点发送自己已经开始下载某个资源的通知，若其他节点向我们发送get\_peer请求（这个请求里也包含一个infohash值），希望我们将这个节点告诉其他节点，我们将这个infohash加入队列，使其可以被DHT client使用。但是爬虫中不能用announce\_peer，因为这就相当于通报虚假资源，对方很容易从上下文中判断你是否通报了虚假资源从而把你禁掉。

在一个DHT网络中当爬虫并不容易，不像普通爬虫一样，看到资源就可以主动爬下来，相反，因为得到资源的方式(get\_peers, announce\_peer)都是被动的，所以爬虫的方式就有些变化了，爬虫所要做的事就是像个正常节点一样去响应其他节点的查询，并且得到其他节点的回应，把其中的数据收集下来就算是完成工作了。而爬虫唯一能做的，是尽可能的去多认识其他节点，这样，才能有更多其他节点来向你询问。这样的话，我们的DHT网络爬虫使用构造虚拟节点、主动路由注入、加入DHT网络并广泛交友的方式实现。

* 构造虚拟节点

Kademlia算法是DHT网络的一种实现。Kademlia的主要目的是用于查询某个资源对应的peer列表，而这个peer列表实际上是分散在整个DHT网络中。DHT网络中节点数量很大，如果要获得peer列表，最简单的做法无非就是依次询问网络中的每个节点。这当然不可行。所以在Kademlia算法中，设立了一个路由表。每一个节点都有一份路由表。这个是按照节点之间的距离关系构建出来的。节点之间的距离当然也有特定的算法定义，在Kademlia中通过对两个节点的ID进行异或操作得到。节点的ID和infohash通过相同算法构建，都是20字节长度。节点和infohash之间也有距离关系，实际上表示的是节点和资源的距离关系。于是我们就利用路由表为爬虫构造出一个虚拟节点。

有了这个路由表之后，再通过一个基于距离关系的查找算法，就可以实现不用挨个遍历就找到特定的节点。而查找资源peer这个操作，正是基于节点查找这个过程。路由表有160个桶，称为K桶，这个桶的数量在实现上可以动态增长。每个桶保存有限个元素，例如K取值为8，指的就是这个桶最多保存8个元素。每个元素就是一个节点，节点包含节点ID、地址信息以及peer信息。这些桶可以通过距离值索引得到，即距离值会经过一个hash算法，使其值落到桶的索引范围内。

* 主动路由注入

一开始我们的DHT网络爬虫是不在DHT网络中的，需要其他DHT节点把自己介绍进去，任何一个在DHT中的节点都可以。在这里，我们可以向

router.bittorrent.com:6881

dht.transmissionbt.com:6881

router.utorrent.com:6881

依次发送find\_node请求，请求中应包括虚拟节点ID，爬虫所在节点IP和端口等信息，然后，对方节点响应爬虫节点的find\_node请求，并主动将爬虫节点加入其路由表中，实现路由注入功能。于是我们的爬虫即注入到以上节点的路由表中了，之后我们的爬虫就可以在DHT网络中开始工作了。

* 广泛交友传播

爬虫的虚拟节点信息被注入到以上三个首节点的路由表中，在首节点与其他DHT网络中的节点进行交互式响应的过程中，爬虫节点信息就有可能被注入到与首节点进行交互的节点的路由表之中。当下次有节点向这些节点发出find\_node请求时，我们的爬虫节点就有可能出现在结果列表中，继而会接收到其他节点的find\_node等操作请求，最终实现了虚拟节点被广泛交友传播的目的。

* 获取节点响应

当爬虫程序加入DHT网络后，它总会收到其他节点发来的announce\_peer消息。announce\_peer消息与get\_peer消息里都带了资源的infohash，但是get\_peer里的infohash对应的资源在该网络中不一定存在，即该资源没有任何可用peer。而announce\_peer则表示已经确认了该网络中有节点正在下载该资源，也即该资源的数据确实存在该网络中。

所以，爬虫程序需要尽最大努力地获取其他节点发来的announce\_peer消息。如果announce\_peer消息会发送给离消息发送节点较近的节点，那么，一方面，爬虫程序应该将自己告诉网络中尽可能多的节点。这可以通过一次完整的find\_node操作实现。另一方面，爬虫程序内部实现可以部署多个DHT节点，总之目的在于尽可能地让爬虫程序成为其他节点的较近者。

* 获取资源信息

获得资源描述信息，其实就是通过infohash获得对应的种子文件。这需要实现P2P协议里的文件分享协议。种子文件的获取其实就是一个文件下载过程，下载到种子文件之后，就可以获取到资源描述。这个过程一种简单的方法，就是从infohash构建出一个磁力链接，然后交给一个支持磁力下载的程序下载种子。从infohash构建出磁力链接非常简单，只需要将infohash编码成磁力链接的xt字段即可，即给infohash加上磁力头：

magnet:?xt=urn:btih:

例如：

文件名：ubuntu-12.04-desktop-i386.iso

Infohash：49FBD26322960D982DA855C54E36DF19AD3113B8

磁力链接：magnet:?xt=urn:btih: 49FBD26322960D982DA855C54E36DF19AD3113B8

* 种子文件分析

torrent文件是一个文本文件，包含了tracker信息和文件信息两部分。tracker信息主要是指BT下载中需要用到的tracker服务器的地址和针对tracker服务器的设置；文件信息是指将目标文件计算处理后再根据BT协议的B编码规则编码后得到的信息，包括BT的主要原理是把提供下载的文件虚拟分成大小相等的块，块大小必须为2Kbyte的整数次方，并把每个块的索引信息和Hash验证码写入torrent文件中，所以torrent文件就是被下载文件的“索引”。

libtorrent是一个功能完整的C++的BitTorrent客户端实现，它运行在嵌入式设备和台式机上。它拥有一个易于使用的文档化库接口，将meta文件生成，hash生成，下载文件, 网络连接等操作封装成接口，供开发者调用，并且都有很好的性能与稳定性。我们在服务器上部署了libtorrent 的python binding，并通过库函调用编程实现了种子文件的解析工作。

# 第四章 系统测试

## 4.1 引言

### 4.1.1 编写目的

编写测试文档的目的是通过整个测试过程，包括测试计划、测试设计、案例编写、测试总结等步骤去了解基于DHT网络和搜索引擎的不良信息处理系统的质量如何。是否存在缺陷，若存在缺陷则其原因是什么以及该如何修复。希望在本系统发布之前能将缺陷修复。同时向用户呈现本系统的具体测试细节，使用户能够明白系统的软件缺陷以及具体的操作方法，这将便于用户理解和使用本系统。本文档面向的读者主要是项目的开发人员和测试人员。

### 4.1.2 开发单位

Eye-guard团队，团队由以下成员组成：

张宇淇（队长），张涛森，叶昌健，于凯齐

## 4.2 评价标准

### 4.2.1 范围

本测试案例采用动态黑盒测试[[[1]](#footnote-1)]的方式，根据现有的需求规格说明书进行测试案例的设计和选择，仅适用于对系统的正确性测试。测试案例将针对系统的核心功能进行测试，主要包括DHT公网嗅探，Torrent种子解析等。

### 4.2.2 数据整理

为了把测试数据加工成便于评价的适当形式，使得测试结果可以同已知结果进行比较。本测试采用手工的方式进行数据的整理，撰写测试报告和统计结果，便于测试人员与程序开发人员之间的交流，也使客户更加直观的理解和查看软件的测试结果。

### 4.2.3 尺度

1. 系统正确的完成了所有需求规格说明书规定的功能；
2. 测试输出结果与预期输出结果之间仅允许有细微的界面偏差，输入、输出的数据必须准确。

## 4.3 计划

### 4.3.1 软件说明

|  |  |  |  |
| --- | --- | --- | --- |
| **功能编号** | **功能名称** | **输入** | **输出** |
| **客户端功能** | | | |
| **TSC-01-01** | 用户成功登录 | 用户名，密码 | 成功进入系统主页 |
| **TSC-01-02** | 用户非法登录 | 用户名，密码 | 显示非法登陆的错误信息 |
| **TSC-01-03** | 查看热门不良资源 | 1. 进入系统首页  2. 点击热门不良资源 | 跳转到近期热门不良资源页面，显示近期不良资源图表 |
| **TSC-01-04** | 报表导出 | 1. 进入系统主页  2. 点击报表导出按钮  3. 选择报表导出项目  4. 选择报表导出格式  5. 确认导出 | 浏览器开始下载对应的报表 |
| **TSC-01-05** | 查看不良结点地理分布 | 1. 进入系统首页  2. 进入DHT网络热点页面 | 跳转到不良结点地理位置分布页面，显示地理位置分布图 |
| **TSC-01-06** | 对不良区域的拓扑结构进行查看 | 1. 进入DHT网络热点图页  2. 点击地图中的不良区域集合点 | 显示该地理区域的DHT网络的拓扑结构 |
| **TSC-01-07** | 查看实时资源分析列表 | 1. 进入系统主页  2. 点击实时资源分析 | 跳转到实时资源分析页面，显示不良资源与不良直播列表 |
| **TSC-01-08** | 对不良资源按照地区选择进行查看 | 1. 进入实时资源分析页  2. 在地区按钮处输入地区过滤关键字 | 不良资源列表根据地区关键字进行显示 |
| **TSC-01-09** | 按照资源类别对不良资源进行过滤 | 1. 进入系统主页  2. 在操作栏选择类别词 | 筛选出不良资源列表中与类别词相关的不良资源 |
| **TSC-01-10** | 按照不良资源捕获时间顺逆序查看 | 1. 进入资源屏蔽历史页  2. 在控制栏点击时间排序按钮 | 不良资源和结点按照时间顺序或者逆序进行排序 |
| **TSC-01-11** | 查看屏蔽历史 | 1. 进入系统首页  2. 点击屏蔽历史按钮 | 跳转到屏蔽历史页面，显示不良资源和不良结点被屏蔽的历史 |
| **TSC-01-12** | 修改系统设置 | 1. 进入系统设置页  2. 修改系统设置  3. 点击确认 | 系统按照系统设置项正常运行 |
| **TSC-01-13** | 不良直播举报 | 1. 进入直播举报页  2. | 被选中的不良直播信息将会反馈到对应的直播平台 |
| **TSC-01-14** | 进入不良资源屏蔽控制台 | 1. 进入系统首页  2. 点击不良资源控制台按钮 | 跳转到资源屏蔽控台页面，显示不良资源列表 |
| **TSC-01-15** | 不良资源屏蔽 | 1. 进入不良资源控制台  2. 点击不良资源项的标记按钮  3. 点击屏蔽按钮 | 被选中的不良资源将会被屏蔽，DHT网络中的结点将无法下载该资源 |
| **TSC-01-16** | 进入不良结点屏蔽控台 | 1. 进入系统首页  2. 点击不良结点控制台按钮 | 跳转到资源屏蔽控台页面，显示不良结点列表 |
| **TSC-01-17** | 不良结点屏蔽 | 1. 进入不良结点控制台  2. 点击不良结点项的标记按钮  3. 点击屏蔽按钮 | 被选中的不良资源将会被屏蔽，DHT网络中该结点中的资源将会无法被下载 |
| **服务器功能** | | | |
| **TSS-02-01** | 添加系统用户 | 1. 打开服务终端  2. 输入添加用户命令  3. 输入新用户的用户名和密码 | 用户被添加 |
| **TSS-02-02** | DHT网络爬虫功能测试 | 1. 打开服务终端  2. 输入运行爬虫命令 | 显示打开rpc server，并且实时滚动获取到的信息 |
| **DT-02-03** | DHT网络资源入库测试 | 1. 打开服务终端  2. 输入数据库入库命令 | 数据成功进入数据库,并在屏幕上滚动显示 |
| **DT-02-04** | 数据库是否正常写入 | 1. 打开数据库终端  2. 输入数据库查询命令 | 数据库内能正常显示入库信息 |
| **DT-02-05** | 后台能否查看爬取信息 | 1. 后台端口已打开  2. 登录 | 能进入后台，且可以正常进行数据显示。 |
| **DT-02-06** | 添加用户 | 1. 连接服务器  2. 输入添加用户命令 | 用户被添加 |
| **DT-02-07** | 获取DHT资源列表API | 提供资源过滤条件（例如所属节点IP，资源类别，捕获时间等） | 返回对应资源列表 |
| **DT-02-08** | 获取DHT节点API | 提供节点过滤条件（例如节点所属地区等） | 返回对应节点 |
| **DT-02-9** | 获取DHT资源详细信息API | 提供资源ID | 返回对应资源的详细信息 |

表 4-1 软件测试计划说明

### 4.3.2 测试目的和要求

#### 4.3.2.1 测试目的

1. 在产品达到预期目的，是否实现项目设计要求；
2. 产品可以稳定运行，并易于操作；
3. 系统产品 Bug 在允许范围之内；
4. 缺陷率在可控制范围内。

#### 4.3.2.2 测试要求

1. 以 80/20 原理为指导，尽量做到在有限的时间里发现尽可能多的缺陷，尤其是严重缺陷；
2. 通过测试，不断完善测试用例；
3. 根据事先定义的测试执行顺序进行测试，并填写测试记录表，保证测试过程是受控制的；
4. 测试重点放在各子系统的功能实现上，问题较多的是按键选择性问题。

### 4.3.3 团队责任

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **任务** | **张宇淇** | **张涛森** | **叶昌健** | **于凯齐** |
| **制定统一的测试方案** |  |  |  |  |
| **制作测试总体进度** |  |  |  |  |
| **制定测试的总体设计** |  |  |  |  |
| **编写测试案例说明** |  |  |  |  |
| **客户端测试** |  |  |  |  |
| **服务器端测试** |  |  |  |  |
| **动态黑盒测试** |  |  |  |  |
| **缺陷报告跟踪与统计** |  |  |  |  |
| **回归测试** |  |  |  |  |

### 4.3.4 测试方法

#### 4.3.4.1 动态黑盒测试

|  |  |
| --- | --- |
| **测试要求** | **描述** |
| **测试目标** | 该项目执行过程中的实际行为能力 |
| **测试范围** | 功能性测试和数据性测试 |
| **技术** | 1. 确认软件至少能做什么，即通过性测试  2. 通过各种手段进行失效性测试 |
| **开始标准** | 项目正常运行 |
| **完成标准** | 项目功能正常显示。 |
| **测试重点和优先级** | 1. 通过性测试  2. 失效性测试  3. 数据测试 |
| **需考虑的特殊事项** | 无 |

### 4.3.5 测试策略

1. 测试计划与需求制定、用例设计同步进行；
2. 必须制定测试需求：通过确定要测试的内容和各自的优先级、重要性，使测试设计工作更有目的性，在需求的指导下设计出更多更有效的用例；
3. 逐步完善测试用例库：测试用例库的建设是一个不断完善的过程，我们要在有限的时间里，先设计出一整套的测试用例，重要的部分用例需要设计得完善一些，一般部分的则指出测试的要点，在以后的测试工作中再不断去完善测试用例库；
4. 测试过程要受到控制：根据事先定义的测试执行顺序进行测试，并填写测试记录表，保证测试过程是受控的；
5. 采用人工测试对软件功能进行动态黑盒测试。

### 4.3.6 测试任务与进度

|  |  |  |  |
| --- | --- | --- | --- |
| **测试活动** | **计划开始日期** | **实际开始日期** | **结束日期** |
| **制定测试计划** | 2018/04/20 | 2018/04/20 | 2018/04/21 |
| **制定测试方案** | 2018/04/20 | 2018/04/20 | 2018/04/22 |
| **功能测试** | 2018/04/23 | 2018/04/23 | 2018/05/23 |
| **测试结果评估** | 2018/05/25 | 2018/05/25 | 2018/05/28 |
| **测试缺陷追踪** | 2018/05/25 | 2018/05/25 | 2018/05/28 |
| **回归测试** | 2018/05/29 | 2018/05/29 | 2018/06/02 |
| **项目提交** | 2018/06/05 | 2018/06/05 | 2018/06/05 |

表 4-4 测试任务与进度

### 4.3.7 风险管理

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **风险描述** | **严重度** | **缓解措施** |
| **1** | 系统不稳定或卡死 | 9 | 找出系统不稳定原因，以及耗损内存模块，进行代码优化。 |
| **2** | 功能不完善，没有完成需求规定功能 | 9 | 确定没有完善的功能，然后进行功能代码编写。 |
| **3** | 模块集成出错导致功能无法实现，或功能出现紊乱 | 8 | 针对某一模块进行修改，加工与测试，然后对几个模块进行集成测试，确认功能。 |
| **4** | 修改某模块时导致其它功能不能使用 | 8 | 代码检查，一般逻辑错误，或代码还原，恢复修改前状态。 |
| **5** | 核心技术无法攻破，导致项目滞后 | 7 | 选用可实现的技术，降低风险，请教指导老师。 |
| **6** | 软件核心开发人员有事调离团队 | 7 | 团队任务分派时，指定两个或两个以上成员担任开发人员。 |
| **7** | 在项目后期中发生软件需求变更 | 9 | 项目设计时，应使项目的架构具有良好可扩展性和模块性。 |
| **8** | 系统出现一些小bug | 5 | 对于不影响系统正常运行的bug可以不处理，锁定bug的原因，对相应代码进行优化。 |
| **9** | 界面有小瑕疵 | 3 | 在时间允许的情况下，可以进行界面优化。 |

## 4.4 测试环境

### 4.4.1 硬件环境

|  |  |
| --- | --- |
| **HW-01/ HW-02/ HW-03** | **环境参数** |
| **硬件设备** | 华硕 X556UB |
| **处理器** | 英特尔 Core i7-6500U @ 2.50GHz 双核 |
| **内存** | 8 GB ( 海力士 DDR3L 1600MHz ) |
| **硬盘** | 120GB SSD + 1TB HDD |
| **网络适配器** | 瑞昱 RTL8168/8111/8112 Ethernet Controller |

表4-7 硬件环境表

### 4.4.2 软件环境

|  |  |
| --- | --- |
| **SW-01** | **环境参数** |
| **操作系统** | Windows 10 |
| **数据库** | Mysql 5.7 |
| **Python 环境** | Python 3.6 + Python 2.7 |
| **Web 浏览器** | Chrome |
| **服务器IP 地址及端口号** | 118.24.64.14:3306 |

表4-8软件环境表

## 4.5 测试案例设计

### 4.5.1 客户端测试案例设计

#### 4.5.1.1 客户端功能测试案例一

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-01** |
| **测试用例名称** | 用户成功登陆 |
| **测试用例说明** | 验证用户成功登录系统 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 打开浏览器  2. 输入localhost:3000，连接到服务器，显示登录界面  3. 输入正确的用户名和密码 |
| **预期结果** | 用户验证成功，进入系统主页 |
| **实际结果** | 实际结果与预期结果一致    图 4-1 系统登录界面    图 4-2 进入系统主页 |
| **结果分析** | 测试通过 |

表4-9 客户端功能测试案例一

#### 4.5.1.2 客户端功能测试案例二

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-02** |
| **测试用例名称** | 用户非法登陆 |
| **测试用例说明** | 验证用户非法登录系统 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 打开浏览器  2. 输入localhost:3000，连接到服务器，显示登录界面  3. 输入错误的用户名和密码 |
| **预期结果** | 无法登陆系统，显示错误登陆信息 |
| **实际结果** | 实际结果与预期结果一致    图 4-3 系统登录界面 |
| **结果分析** | 测试通过 |

表4-10 客户端功能测试案例二

#### 4.5.1.3 客户端功能测试案例三

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-03** |
| **测试用例名称** | 查看热门不良资源 |
| **测试用例说明** | 验证不良资源是否按照下载次数降序显示 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击热门不良资源 |
| **预期结果** | 跳转到近期热门不良资源页面，显示近期不良资源图表 |
| **实际结果** | 实际结果与预期结果一致    图 4-4 热门不良资源页 |
| **结果分析** | 测试通过 |

表4-11 客户端功能测试案例三

#### 4.5.1.4 客户端功能测试案例四

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-04** |
| **测试用例名称** | 报表导出 |
| **测试用例说明** | 验证系统是否能到处数据 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击热门报表导出按钮  3. 选择报表导出项目  4. 选择报表打出格式  5. 确认导出 |
| **预期结果** | 浏览器开始下载对应的报表 |
| **实际结果** | 实际结果与预期结果一致    图 4-5 报表导出页 |
| **结果分析** | 测试通过 |

表4-12 客户端功能测试案例四

#### 4.5.1.5 客户端功能测试案例五

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-05** |
| **测试用例名称** | 查看不良节点地理分布 |
| **测试用例说明** | 验证系统能生成不良结点地理分布图 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击DHT网络热点图 |
| **预期结果** | 跳转到近期热门不良结点地理位置页面，显示地理位置分布图 |
| **实际结果** | 实际结果与预期结果一致    图 4-6 DHT网络热点图页 |
| **结果分析** | 测试通过 |

表4-13 客户端功能测试案例五

#### 4.5.1.6 客户端功能测试案例六

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-06** |
| **测试用例名称** | 对不良区域的拓扑结构进行查看 |
| **测试用例说明** | 验证系统能生成不良区域的拓扑结构 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入DHT网络热点图页  2. 点击地图中的不良区域集合点 |
| **预期结果** | 显示该地理区域的DHT网络的拓扑结构 |
| **实际结果** | 实际结果与预期结果一致    图 4-7 拓扑结构页 |
| **结果分析** | 测试通过 |

表4-14 客户端功能测试案例六

#### 4.5.1.7 客户端功能测试案例七

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-07** |
| **测试用例名称** | 查看实时资源分析列表 |
| **测试用例说明** | 验证系统能否正确加载实时资源分析列表 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统主页  2. 点击资源实时分析按钮 |
| **预期结果** | 跳转到资源实时分析页面，显示不良资源和不良直播列表 |
| **实际结果** | 实际结果与预期结果一致    图 4-8 实时资源分析界面 |
| **结果分析** | 测试通过 |

表4-15 客户端功能测试案例七

#### 4.5.1.8 客户端功能测试案例八

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-08** |
| **测试用例名称** | 对不良资源按照地区选择进行查看 |
| **测试用例说明** | 验证系统能否按照关键词对不良资源进行过滤 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入资源实时分析页  2. 在地区按钮处下拉选择地区过滤关键字 |
| **预期结果** | 不良资源列表根据地区关键字进行显示 |
| **实际结果** | 实际结果与预期结果一致      图 4-9 地区过滤列页 |
| **结果分析** | 测试通过 |

表4-16 客户端功能测试案例八

#### 4.5.1.9 客户端功能测试案例九

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-09** |
| **测试用例名称** | 按照资源类别对不良资源进行过滤 |
| **测试用例说明** | 验证系统能否按照关键词对不良资源进行过滤 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入资源实时分析页  2. 在操作栏选择类别词 |
| **预期结果** | 筛选出不良资源列表中与类别词相关的不良资源 |
| **实际结果** | 实际结果与预期结果一致      图 4-10 类别过滤页 |
| **结果分析** | 测试通过 |

表4-17 客户端功能测试案例九

#### 4.5.1.10 客户端功能测试案例十

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-10** |
| **测试用例名称** | 按照不良资源的捕获时间逆序查看 |
| **测试用例说明** | 验证系统能否对不良资源按时间进行排序 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统主页  2. 在操作栏选择时间排序按钮 |
| **预期结果** | 不良资源和结点按照时间逆序进行排序 |
| **实际结果** | 实际结果与预期结果一致    图 4-11 按时间排序结果页 |
| **结果分析** | 测试通过 |

表4-18 客户端功能测试案例十

#### 4.5.1.11 客户端功能测试案例十一

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-11** |
| **测试用例名称** | 查看屏蔽历史 |
| **测试用例说明** | 验证系统能否加载屏蔽历史 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击屏蔽历史按钮 |
| **预期结果** | 跳转到屏蔽历史页，显示被不良资源和不良结点被屏蔽的历史 |
| **实际结果** | 实际结果与预期结果一致    图 4-12 屏蔽历史页 |
| **结果分析** | 测试通过 |

表4-19 客户端功能测试案例十一

#### 4.5.1.12 客户端功能测试案例十二

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-12** |
| **测试用例名称** | 修改系统设置 |
| **测试用例说明** | 验证系统能否按照系统设置项正常运行 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统设置页  2. 修改系统设置  3. 点击确认 |
| **预期结果** | 系统按照系统设置项正常运行 |
| **实际结果** | 实际结果与预期结果一致    图 4-13 系统设置页 |
| **结果分析** | 测试通过 |

表4-20 客户端功能测试案例十二

#### 4.5.1.13 客户端功能测试案例十三

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-13** |
| **测试用例名称** | 不直播举报 |
| **测试用例说明** | 验证系统能否对不良直播进行举报 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入不良直播页  2. 点击不良直播项的举报按钮 |
| **预期结果** | 被选中的不良直播信息将会被反馈到对应的直播平台 |
| **实际结果** | 实际结果与预期结果一致    图 4-14 不良直播举报页 |
| **结果分析** | 测试通过 |

表4-21 客户端功能测试案例十三

#### 4.5.1.14 客户端功能测试案例十四

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-14** |
| **测试用例名称** | 进入不良资源屏蔽控制台 |
| **测试用例说明** | 验证系统能否加载不良资源屏蔽控制台 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击不良资源控制台 |
| **预期结果** | 跳转到不良资源控制台页面，显示不良资源列表 |
| **实际结果** | 实际结果与预期结果一致  \  图 4-15资源屏蔽控制台页 |
| **结果分析** | 测试通过 |

表4-22 客户端功能测试案例十四

#### 4.5.1.15 客户端功能测试案例十五

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-15** |
| **测试用例名称** | 不良资源屏蔽 |
| **测试用例说明** | 验证系统能否对不良资源进行屏蔽 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入不良资源控制台  2. 点击不良资源项的标记按钮  3. 点击屏蔽按钮 |
| **预期结果** | 被选中的不良资源将会被屏蔽，DHT网络中的结点将无法下载该资源 |
| **实际结果** | 实际结果与预期结果一致    图 4-16 不良资源屏蔽页 |
| **结果分析** | 测试通过 |

表4-23 客户端功能测试案例十五

#### 4.5.1.16 客户端功能测试案例十六

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-16** |
| **测试用例名称** | 进入不良结点控制台 |
| **测试用例说明** | 验证系统能否加载不良结点屏蔽控制台 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入系统首页  2. 点击不良资源控制台 |
| **预期结果** | 跳转到不良资源控制台页面，显示不良资源列表 |
| **实际结果** | 实际结果与预期结果一致    图 4-17 节点屏蔽控制台页 |
| **结果分析** | 测试通过 |

表4-24 客户端功能测试案例十六

#### 4.5.1.17 客户端功能测试案例十七

|  |  |
| --- | --- |
| **测试用例ID** | **DT-01-17** |
| **测试用例名称** | 不良结点屏蔽 |
| **测试用例说明** | 验证系统能否对不良结点进行屏蔽 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 进入不良结点控制台  2. 点击不良结点项的标记按钮  3. 点击屏蔽按钮 |
| **预期结果** | DHT网络中该结点中的资源将会被无法下载 |
| **实际结果** | 实际结果与预期结果一致    图 4-18 不良结点屏蔽页 |
| **结果分析** | 测试通过 |

表4-25 客户端功能测试案例十七

### 4.5.2 服务端测试案例设计

#### 4.5.2.1服务端模块测试案例一

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-01** |
| **测试用例名称** | 添加系统用户 |
| **测试用例说明** | 验证能否添加系统用户用于监测DHT网络爬虫 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 打开服务终端  2. 输入添加用户命令  3. 输入新用户的用户名和密码 |
| **预期结果** | 终端提示添加用户成功 |
| **实际结果** | 实际结果与预期结果一致    图 4-19 服务终端 |
| **结果分析** | 测试通过 |

表4-26 服务功能测试案例一

#### 4.5.2.2服务端模块测试案例二

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-02** |
| **测试用例名称** | DHT网络爬虫功能测试 |
| **测试用例说明** | 测试网络爬虫能否正常嗅探到DHT网络中的资源 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 打开服务终端  2. 输入运行爬虫命令 |
| **预期结果** | 正常运行，并且实时滚动获取到的信息 |
| **实际结果** | 实际结果与预期结果一致    图 4-20 服务终端 |
| **结果分析** | 测试通过 |

表4-27 服务功能测试案例二

#### 4.5.2.3服务端模块测试案例三

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-03** |
| **测试用例名称** | DHT网络资源入库测试 |
| **测试用例说明** | 验证DHT网络资源能否正常加入数据库 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 1. 打开服务终端  2. 输入数据库入库命令 |
| **预期结果** | 数据成功进入数据库,并在屏幕上滚动显示 |
| **实际结果** | 实际结果与预期结果一致    图 4-21 服务终端 |
| **结果分析** | 测试通过 |

表4-28 服务功能测试案例三

#### 4.5.2.4服务端模块测试案例四

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-04** |
| **测试用例名称** | 数据库是否正常写入 |
| **测试用例说明** | 验证上次写入的数据是否正常写入数据库 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 输入数据库查询命令 |
| **预期结果** | 数据库内能正常显示入库信息 |
| **实际结果** | 实际结果与预期结果一致    图 4-22 数据库端 |
| **结果分析** | 测试通过 |

表4-29 服务功能测试案例四

#### 4.5.2.5服务端模块测试案例五

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-05** |
| **测试用例名称** | 后台能否查看爬取信息 |
| **测试用例说明** | 系统有一个爬虫后台，可以查看爬取的信息,验证该系统能否正常显示 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 后台端口已打开 |
| **预期结果** | 能进入后台，且可以正常进行数据显示。 |
| **实际结果** | 实际结果与预期结果一致，系统正常显示爬取到的资源。    图 4-23 服务器后台 |

表4-30 服务功能测试案例五

#### 4.5.2.6服务端模块测试案例六

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-06** |
| **测试用例名称** | 添加用户 |
| **测试用例说明** | 验证能否添加系统用户 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 用户名，密码，用户邮箱 |
| **预期结果** | 创建用户成功 |
| **实际结果** | 实际结果与预期结果一致，创建用户成功    图 4-24 服务器后台 |

表4-31 服务功能测试案例六

#### 4.5.2.7服务端模块测试案例七

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-07** |
| **测试用例名称** | 获取DHT资源列表 |
| **测试用例说明** | 验证能否获取DHT资源列表 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 资源过滤条件（时间，地区，类型） |
| **预期结果** | 获取到资源列表 |
| **实际结果** | 实际结果与预期结果一致    图 4-25 服务器后台 |

表4-32 服务功能测试案例七

#### 4.5.2.8服务端模块测试案例八

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-08** |
| **测试用例名称** | 获取DHT节点 |
| **测试用例说明** | 验证能否获取DHT节点信息 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 节点IP地址 |
| **预期结果** | 获取节点列表 |
| **实际结果** | 实际结果与预期结果一致    图 4-26 web后台 |

表4-33 服务功能测试案例八

#### 4.5.2.9服务端模块测试案例九

|  |  |
| --- | --- |
| **测试用例ID** | **DT-02-09** |
| **测试用例名称** | 获取DHT资源详细信息 |
| **测试用例说明** | 验证能否获取DHT资源详细信息 |
| **预置条件** | 服务器处于打开状态 |
| **输入** | 资源ID |
| **预期结果** | 获取详细信息成功 |
| **实际结果** | 实际结果与预期结果一致    图 4-27 资源详情 |

表4-34 服务功能测试案例九

1. [] 不深入代码细节的软件测试方法称为动态黑盒子测试。 [↑](#footnote-ref-1)