电类工程导论（C类）

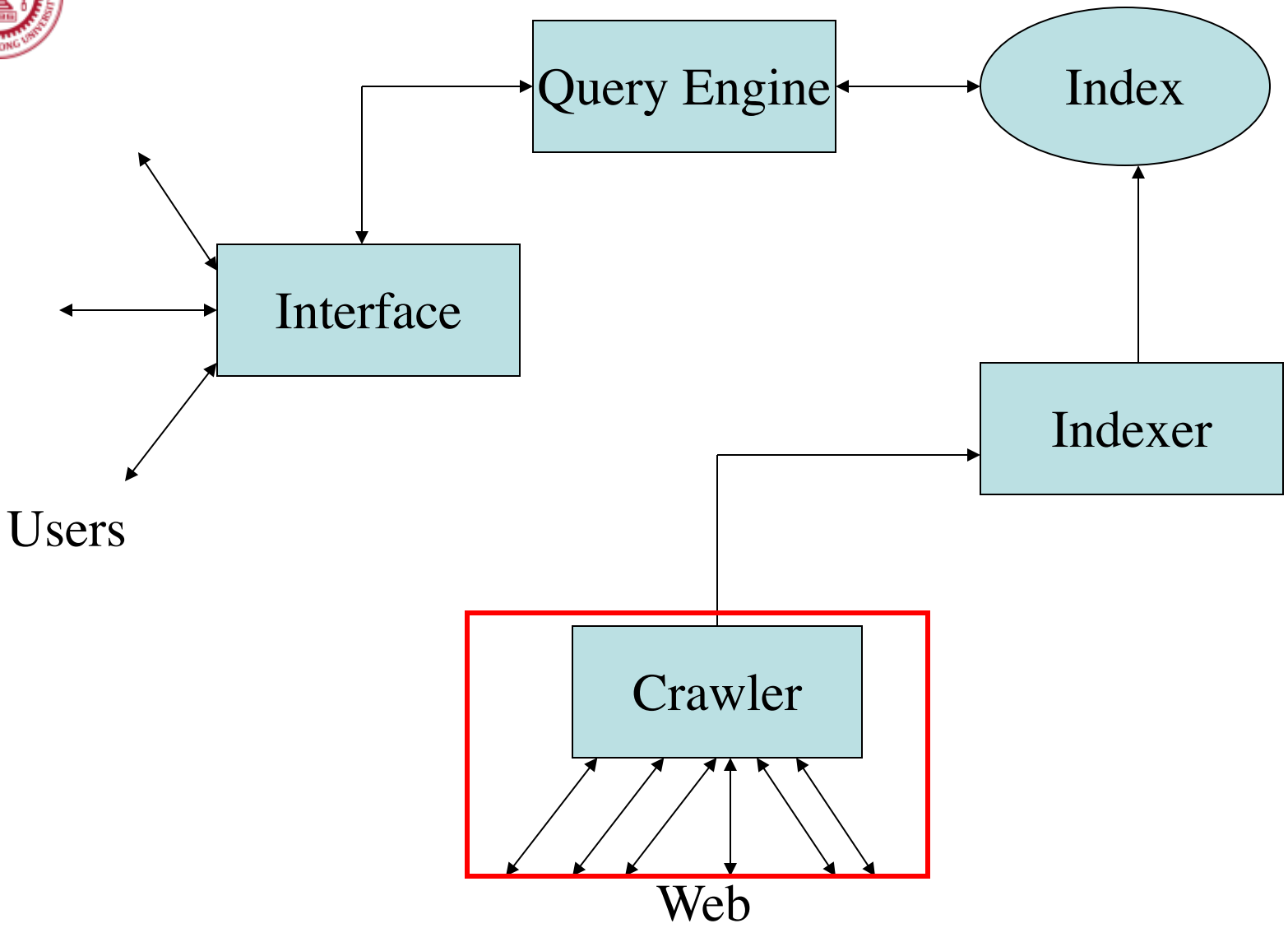
张娅　　何大治

电子信息与电气工程学院

电子工程系

网络爬虫

Crawlers/Spiders/Robots/Bots



A Typical Web Search Engine

网络爬虫

- Web搜索引擎必须要要采集网页文档
- 网络爬虫是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本，是搜索引擎的重要组成部分
- 网络爬虫是搜索引擎中最核心的部分，整个搜索引擎的素材库来源于网络爬虫的采集，网络爬虫的性能好坏直接影响这搜索引擎整体性能和处理速度
- 网络爬虫还要完成信息提取任务
  - 分析提取抓取回来的网页内容：如提取其中的URL
  - 识别MP3、图片、flash等不同内容

抓取对象

- 静态网页
  - 爬虫从一个或若干初始网页的URL开始，获得初始网页上的URL。在抓取网页的过程中，不断从当前页面上抽取新的URL放入队列，直到满足系统的一定停止条件
- 动态网页
  - 分析动态网页参数，按照一定规章，“拼”出所有要被抓取的URL，只抓取这些特定范围内动态网页
- 特殊内容
  - 如RSS、XML数据，情况特殊需特殊处理。如新闻的滚动新闻页面，需要爬虫不停地监控扫描，发现新内容马上就进行抓取
- 文件对象
  - 图片、MP3、Flash、视频等文件的抓取，都要特殊处理。

网络爬虫必须提供的功能特点

礼貌性

- 不要高频率采集某个网站
- 仅仅采集robots.txt所规定的可以采集的网页

鲁棒性

- 能够处理采集器陷阱、重复页面、超大页面、超大网站、动态页面等问题

网络爬虫应该提供的功能特点

- 分布式：能够跨多台机器进行分布式处理
- 规模的可扩展性：能够通过增加机器支持更高的采集速度
- 质量：优先采集高质量网页
- 新鲜度（Freshness）：能够持续运行：对已采集网页进行更新
- 性能和效率：能有效利用包括处理器、存储器、和网络带宽在内的系统资源
- 功能的可扩展性：支持多方面方便地进行功能扩展，如处理新的数据格式、新的抓取协议

基本的采集过程

- 初始化采集URL种子队列；
- 重复如下过程：
  - 从队列中取出URL
  - 下载并分析网页
  - 从网页中抽取更多的URL
  - 将这些URL放到队列中

- 整个采集过程可看做是Web图的遍历过程
- 这里有个“Web的连通性很好”的基本假设

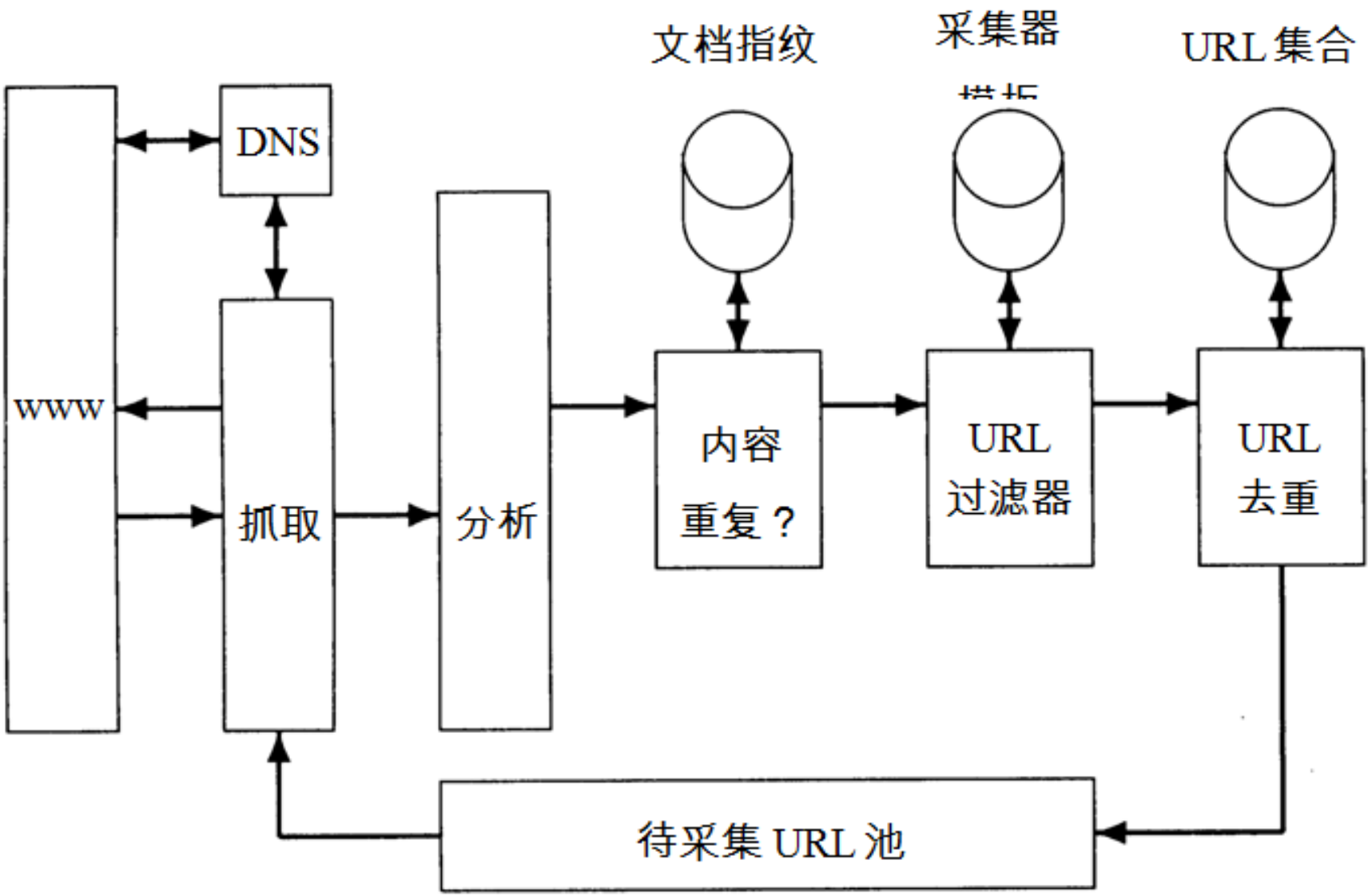
课堂思考题: 下列爬虫有什么问题？

```
urlqueue := (some carefully selected set of seed urls)
while urlqueue is not empty:
        myurl := urlqueue.getlastanddelete()
        mypage := myurl.fetch()
        fetchedurls.add(myurl)
        newurls := mypage.extracturls()
        for myurl in newurls:
                if myurl not in fetchedurls and not in urlqueue:
                 urlqueue.add(myurl)
        addtoinvertedindex(mypage)
```

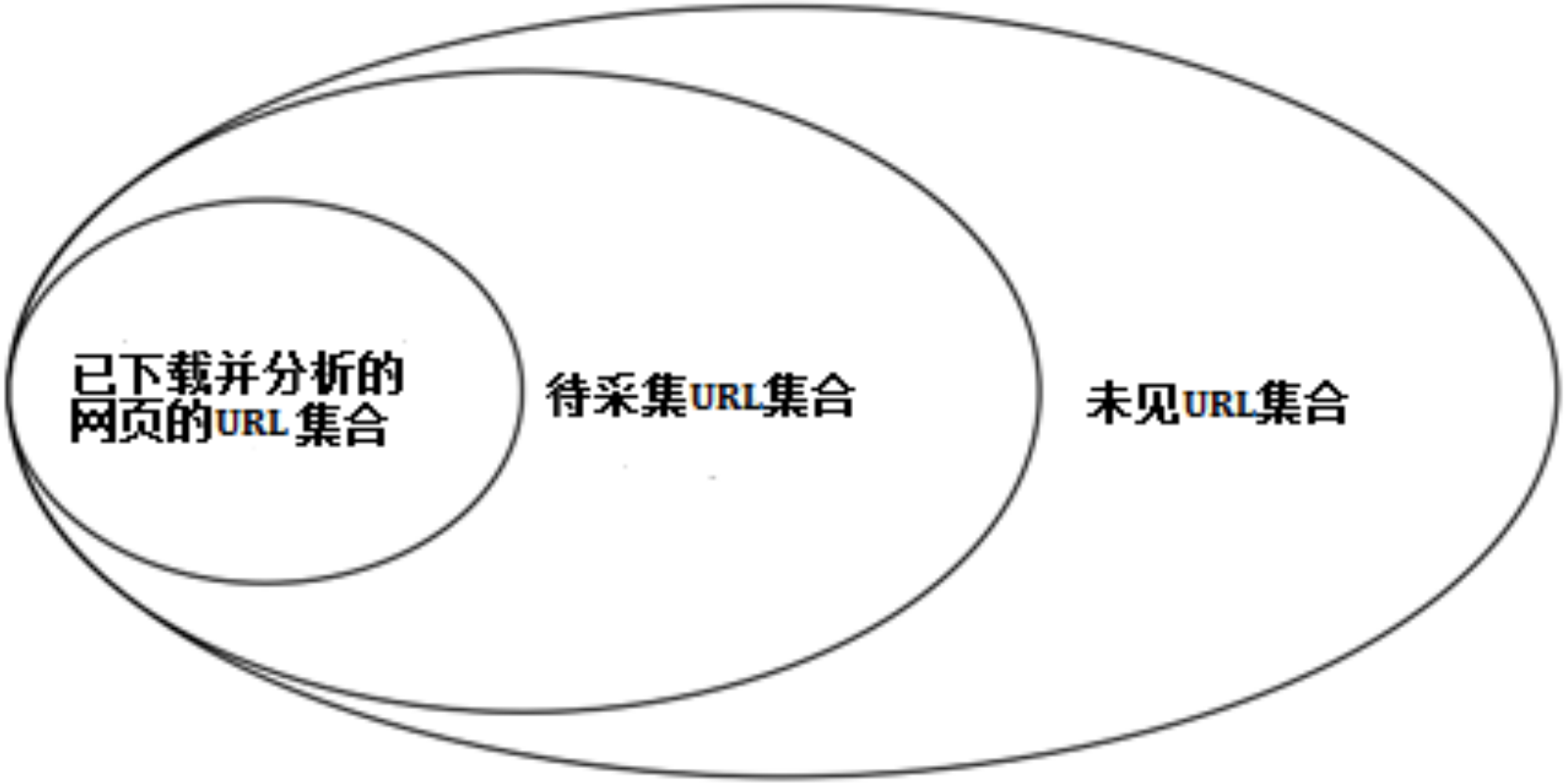
上述简单采集器的问题

- 采集策略：在给定时间内，只可以抓取所下载网络的一部分，需要对它的抓取页面设置优先级
- 重复网页：必须要集成重复检测功能
- 选择性索引：我们不可能索引所有网页，必须要从中选择部分网页，如何选择？
- 作弊网页和采集器陷阱：必须要集成作弊网页检测功能
- 礼貌性问题： 对同一网站的访问按遵照协议规定，并且访问的间隔必须要足够
- 新鲜度(freshness)问题：必须要定期更新或者重采
  - 由于Web的规模巨大，我们只能对一个小的网页子集频繁重采
  - 同样，这也存在一个选择或者优先级问题
- 规模问题: 必须要分布式处理

爬虫基本架构



待采集URL池(URL frontier)

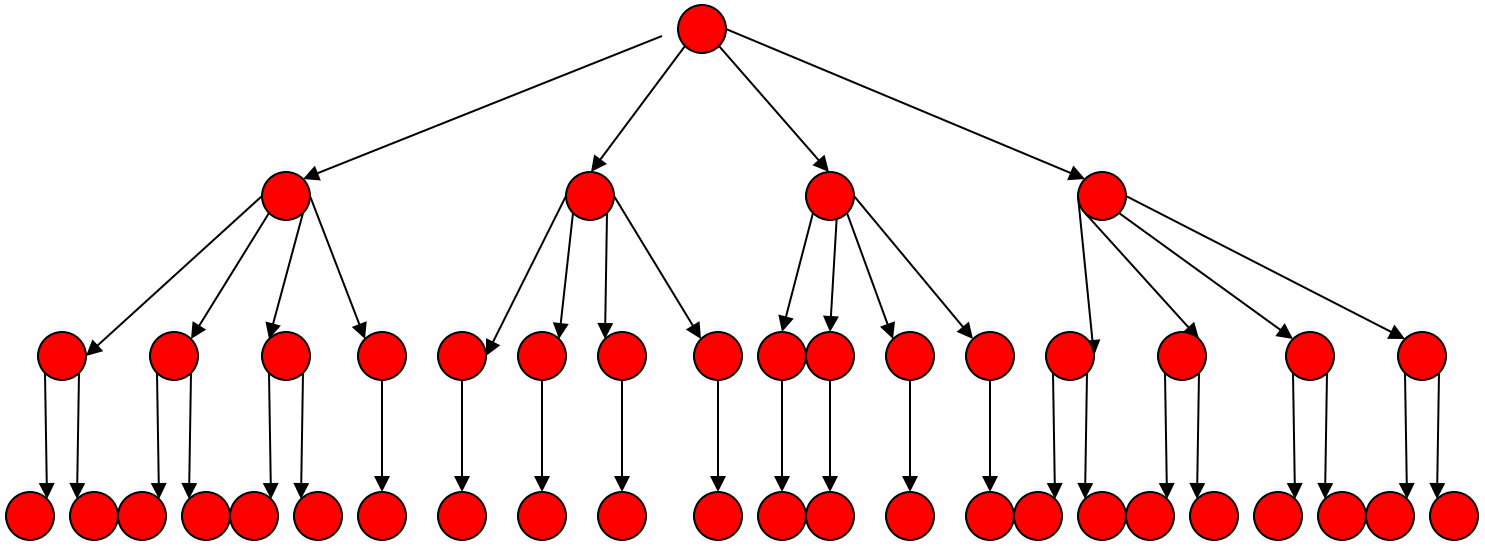


待采集URL池

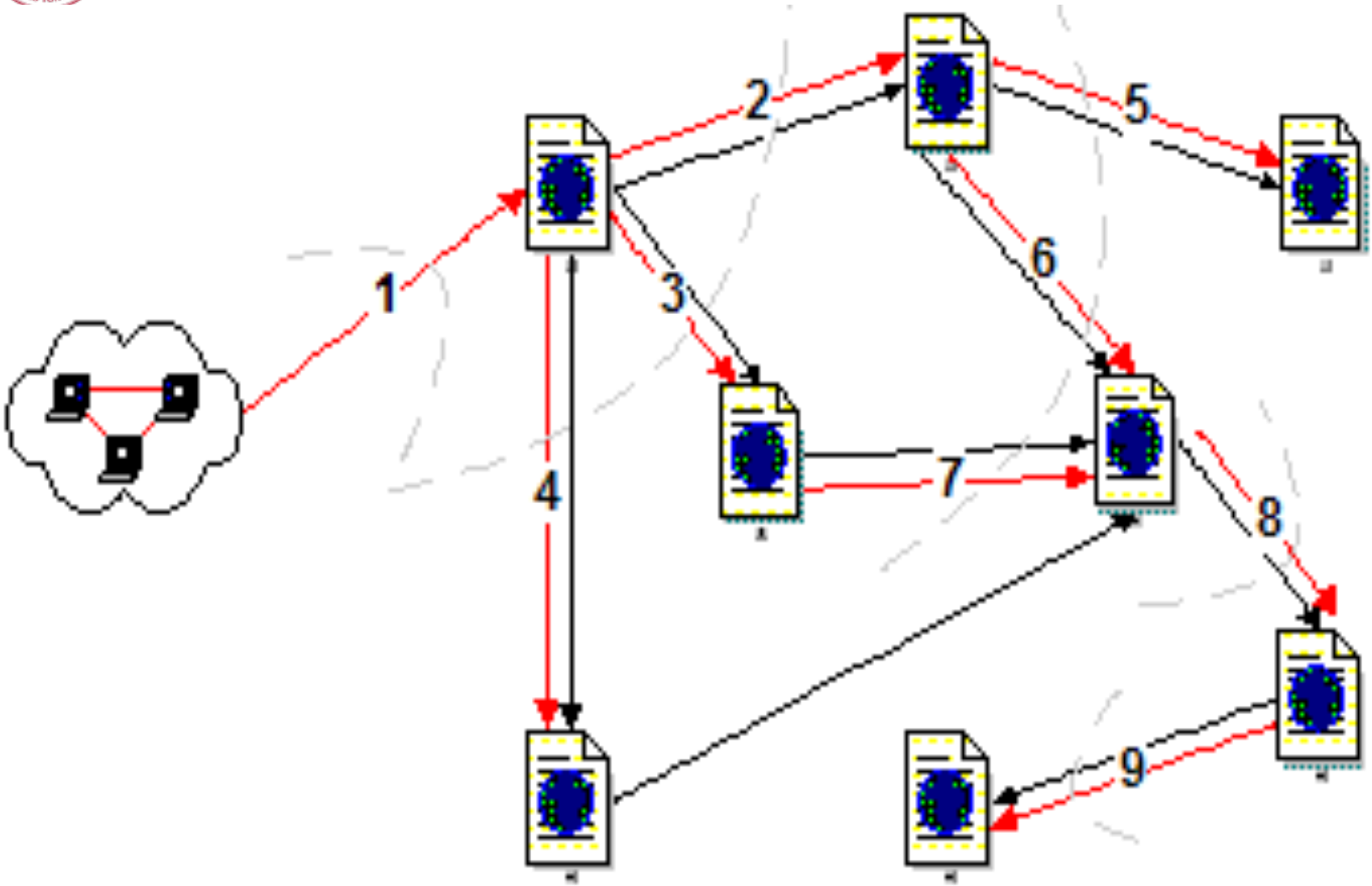
- 待采集URL池是一个数据结构，它存放并管理那些已经看到但是还没有采集的URL集合
- 可能包含来自同一主机的不同页面
- 必要要避免在同一时间采集这些来自同一主机的页面
- 必须要保证采集线程任务饱和

广度优先搜索策略

- 在抓取过程中，在完成当前层次的搜索后，才进行下一层次的搜索。
- 算法设计和实现相对简单。在目前为覆盖尽可能多的网页，多采用广度优先搜索方法
- 也被用于主题爬虫中，因为与初始URL在一定链接距离内的网页具有主题相关性的概率很大

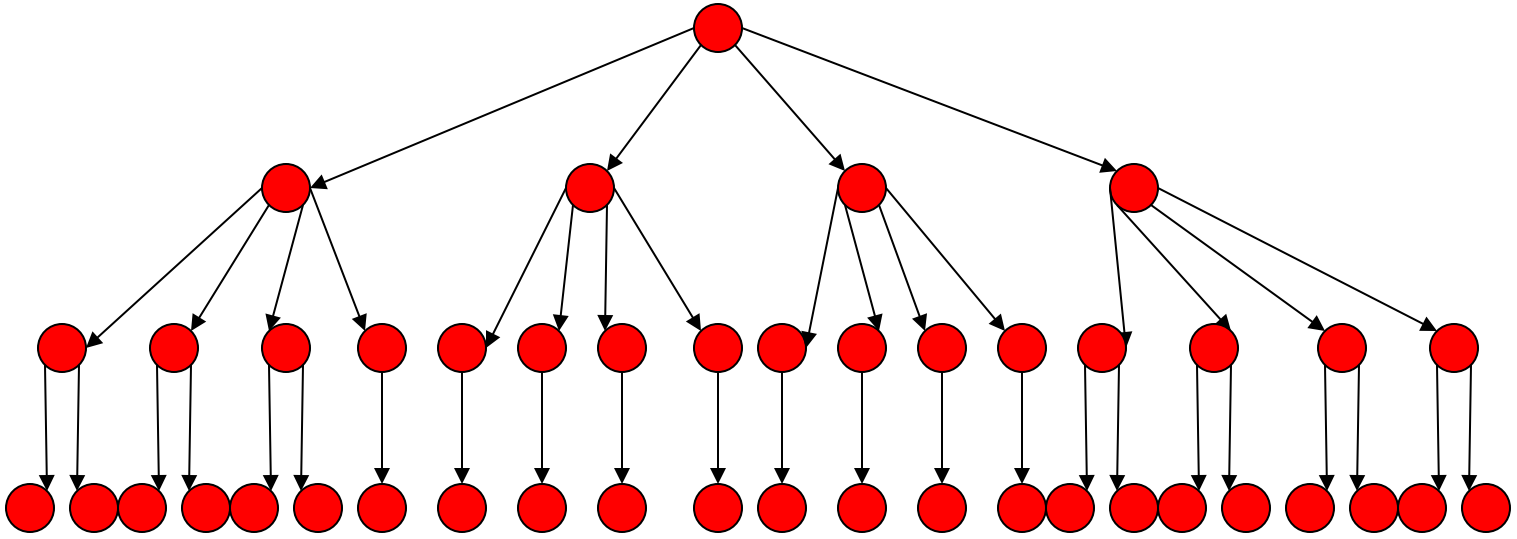


广度优先爬行策略

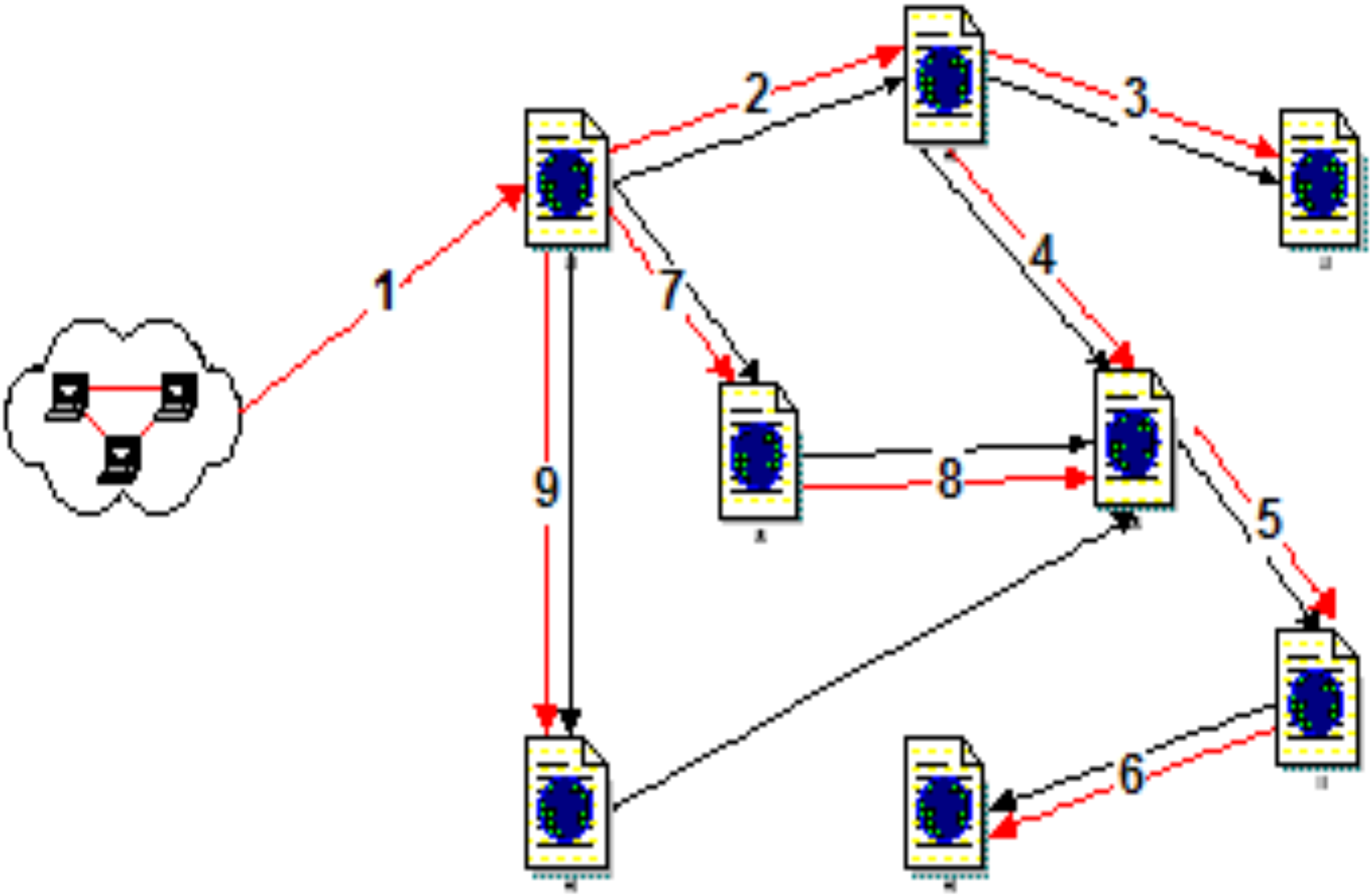


深度优先爬行策略

- 在开发网络爬虫早期使用较多的方法之一，目的是要到达叶节点，即那些不包含任何超链接的页面文件
- 从起始页开始在当前HTML文件中，当一个超链接被选择后，被链接的HTML文件将执行深度优先搜索，一个链接一个链接跟踪下去，处理完这条线路之后再转入下一个起始页，继续跟踪链接。



深度优先爬行策略



避免重复爬取页面

- 避免重复访问已经被爬取过的网页
- 对已经访问过的页面建立有效索引以快速识别
  - Tree indexing (e.g. trie)
  - Hashtable
- 可用URL作为页面索引的关键词（key）
  - 必须实现URL 规范化 (e.g. delete ending “/”)
  - 不能检测重复页面或镜像页面
- 用文本内容作为关键词索引页面
  - 需要首先下载页面

URL 规范化(normalization)

- 在网页分析过程中，必须要将相对URL地址规范化
- 从网页中抽取的URL有些是相对地址
- 比如，在http://mit.edu网站下，我们会采集页面aboutsite.html
  - 该页面的绝对地址为： http://mit.edu/aboutsite.html
- Equivalent variations of ending directory normalized by removing ending slash.
  - http://www.cs.utexas.edu/users/mooney/
  - http://www.cs.utexas.edu/users/mooney
- Internal page fragments (ref’ s) removed:
  - http://www.cs.utexas.edu/users/mooney/welcome.html#courses
  - http://www.cs.utexas.edu/users/mooney/welcome.html

内容重复判别

- 对每个抓取的页面，判断它是否已在索引当中
- 可通过比较两个网页共有的子字符串来衡量两个网页的相似程度
- 可以采用文档指纹或者shingle的方法判别
- 忽略那些已经在索引中的重复页面

Shingle

- Shingling算法（Broder et al, 1997）
  - “w-shingle”：从文档中提取的所有长度为w的子字符串
  - 比较Shingle的重叠程度来衡量网页的相似性
- 文档：To be, or not to be: that is the question
- 分词后的词汇(token，语汇单元)集合是：

  (to, be, or, not, to, be, that, is, the, question)
- 那么w=2的2-shingling就是集合:

  {(be or), (be that), (is the), (not to), (or not), (that is), (the question), (to be)}
  - 实际网页搜索中，w=11

基于Single的网页相似性

- 给定shingle的大小,两个文档A和B的相似度r 取决于它们共有的shingle数量：

$$r(A,B)=\frac{|S(A)\bigcap S(B)|}{|S(A)\bigcup S(B)|}$$

其中|A|表示集合A的大小

- 相似度是介于0和1之间的一个数值，且r(A,A)=1,即一个文档和它自身100%相似。

- 实际计算中，取每个shingle的哈希值

礼貌性

礼貌性

- 不要高频率采集某个网站
- 仅仅采集robots.txt所规定的可以采集的网页

- Robots.txt文件
- 1994年起使用的采集器协议(即规定了采集器对网站的访问限制)
- 采集时，要将每个站点的 robots.txt放到高速缓存中，这一点相当重要

Robot Exclusion Protocol Examples

- Exclude specific directories:

```
User-agent: *
Disallow: /tmp/
Disallow: /cgi-bin/
Disallow: /users/paranoid/
```

- Exclude a specific robot:

```
User-agent: GoogleBot
Disallow: /
```

- Allow a specific robot:

```
User-agent: GoogleBot
Disallow:

User-agent: *
Disallow: /
```

Example of a robots.txt (nih.gov)

```
User-agent: PicoSearch/1.0
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
User-agent: *
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
Disallow: /ddir/
Disallow: /sdminutes/
```

Robots META Tag

- Include META tag in HEAD section of a specific HTML document.
  - <meta name= “robots” content= “none” >
- Content value is a pair of values for two aspects:
  - index | noindex: Allow/disallow indexing of this page.
  - follow | nofollow: Allow/disallow following links on this page.
- META tag is newer and less well-adopted than “robots.txt”. (maybe not used as much)

Robots META Tag (cont)

- Special values:
  - all = index,follow
  - none = noindex,nofollow
- Examples:

<meta name= “robots” content= “noindex,follow” >

<meta name= “robots” content= “index,nofollow” >

<meta name= “robots” content= “none” >

采集规模的数量级

- 如果要在一个月内采集20,000,000,000个页面. . .

. . . 那么必须要在一秒内大概采集 8000个网页!

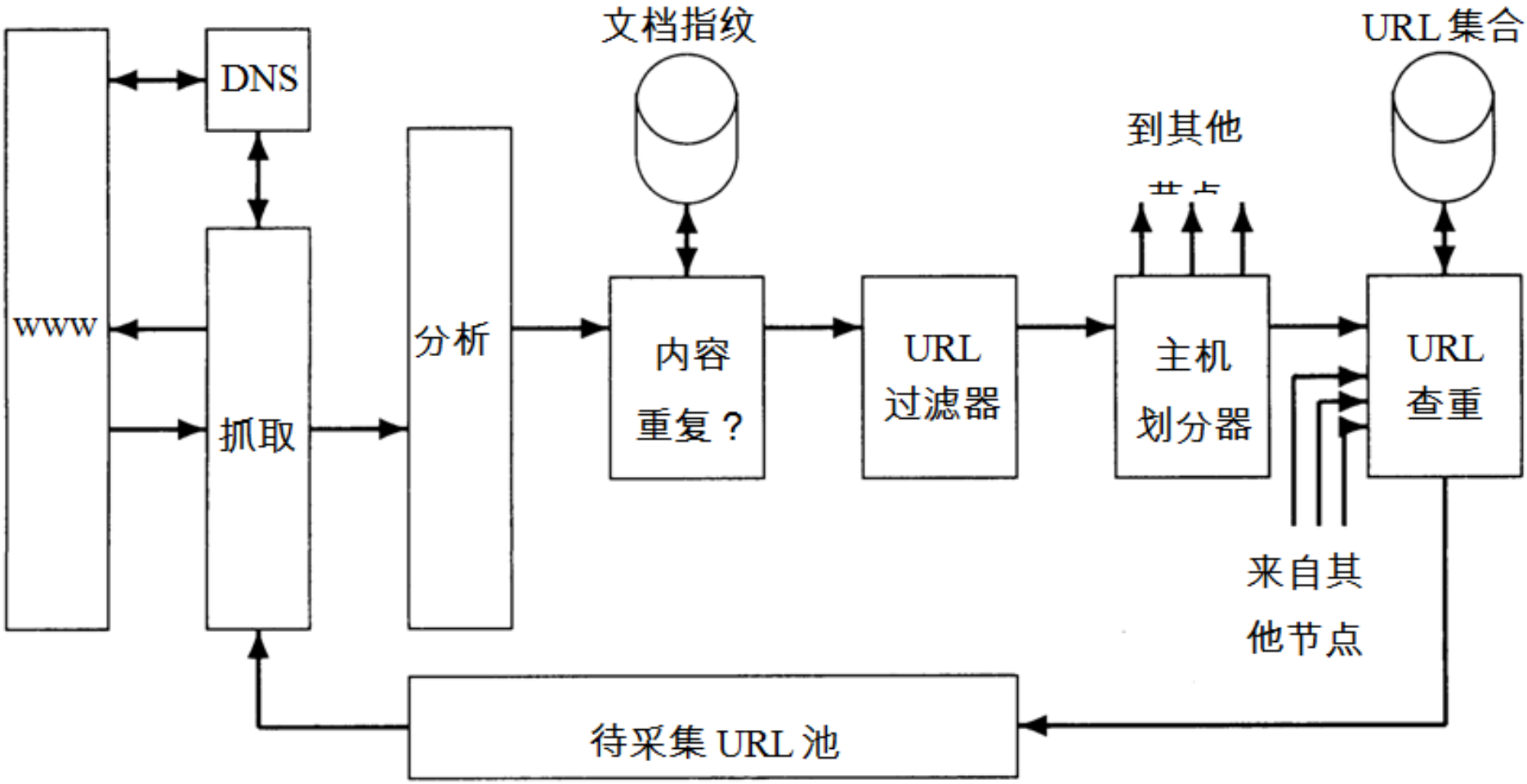
- 由于我们采集的网页可能重复、不可下载或者是作弊网页，实际上可能需要更快的采集速度才能达到上述指标

分布式采集

- 运行多个采集线程，这些线程可以分布在不同节点上
  - 这些节点往往在地理上分散在不同位置
- 将采集的主机分配到不同节点上

Google 数据中心(wazfaring. com)

分布式采集器


文档指纹
URL集合
DNS
到其他
www
分析
内容
重复？
URL
过滤器
主机
划分器
URL
查重
抓取
来自其
他节点
待采集URL池

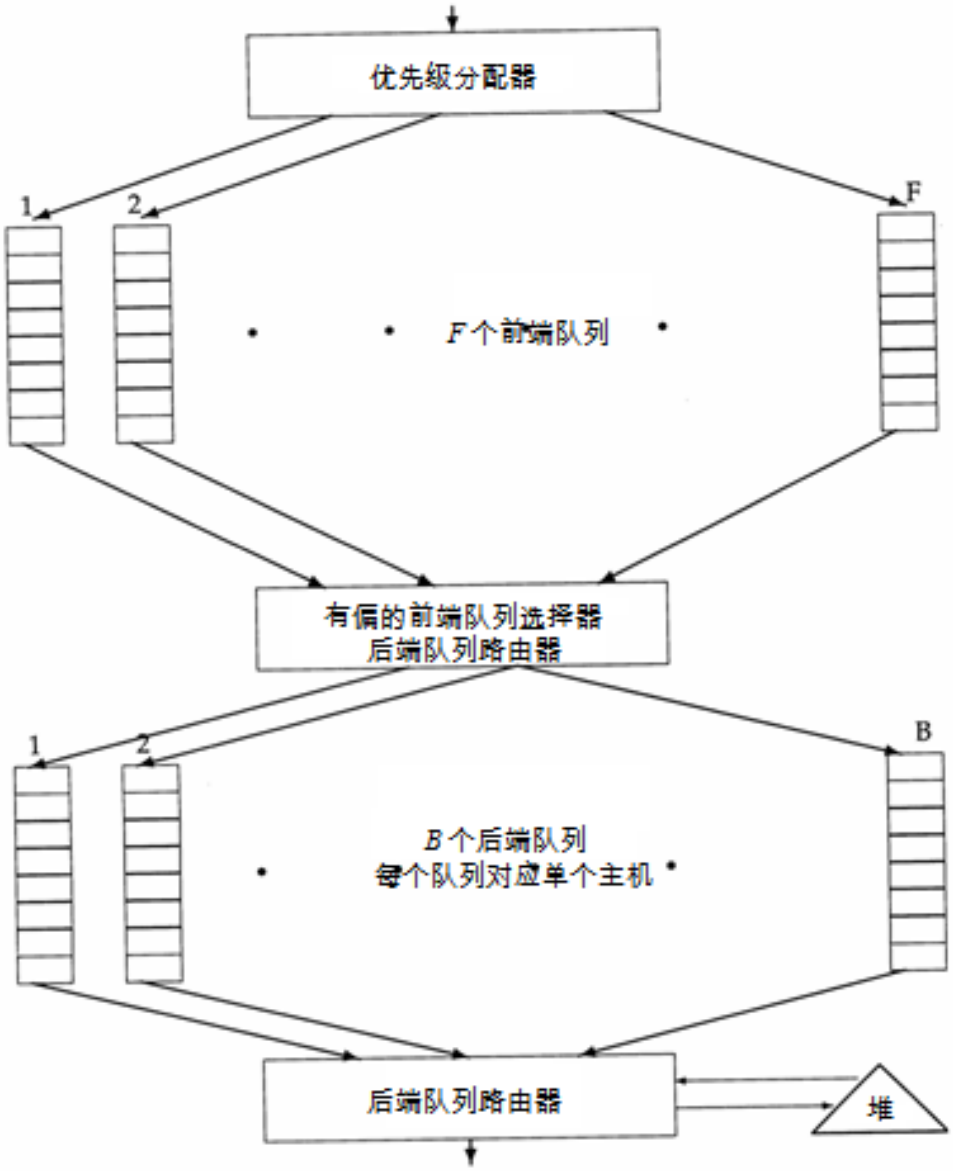
采集池中URL的输出顺序

- 新鲜度:频繁改变的高质量网页优先考虑频繁采集
  - 比如，对某些网站的采集频率(如新闻网站)要高于其他网站
- 礼貌性: 不要非常频繁第访问某个Web服务器
  - 比如，可以在两次服务器访问之间设置一个时间间隔
- 要实现上述功能并不容易，一个简单的优先级队列难以成功

网络爬虫实例
Mercator(Altavista Crawler)

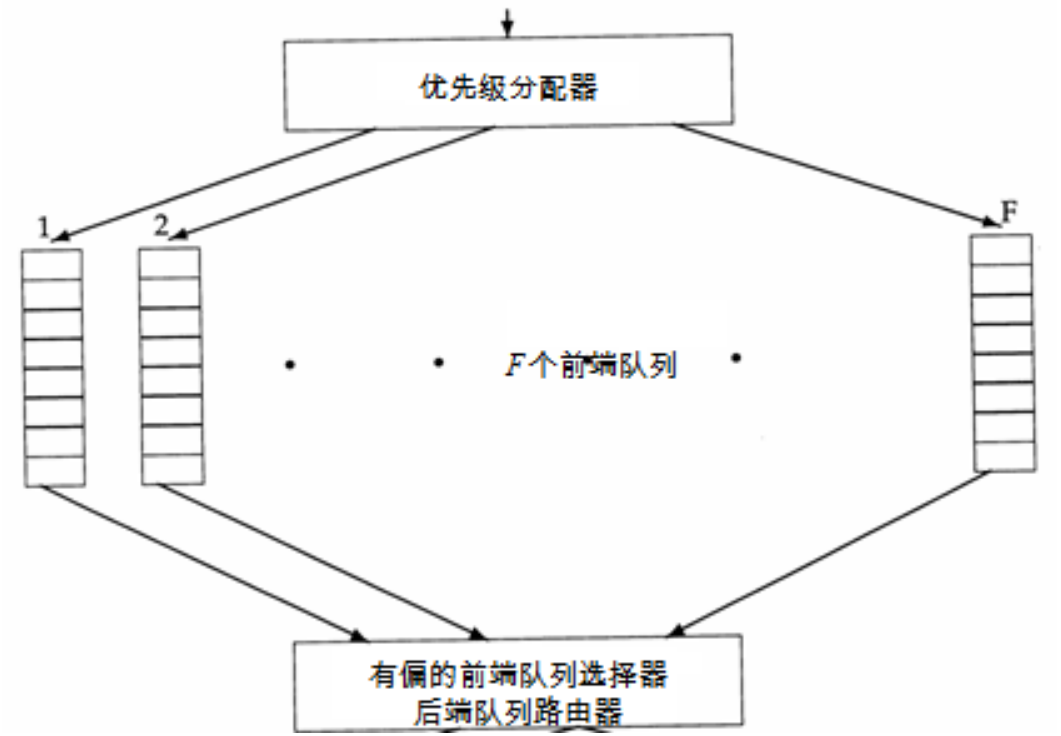
Mercator: A scalable, extensible web crawler (Heydon et al. 1999)

Mercator 中的待采集URL缓冲池



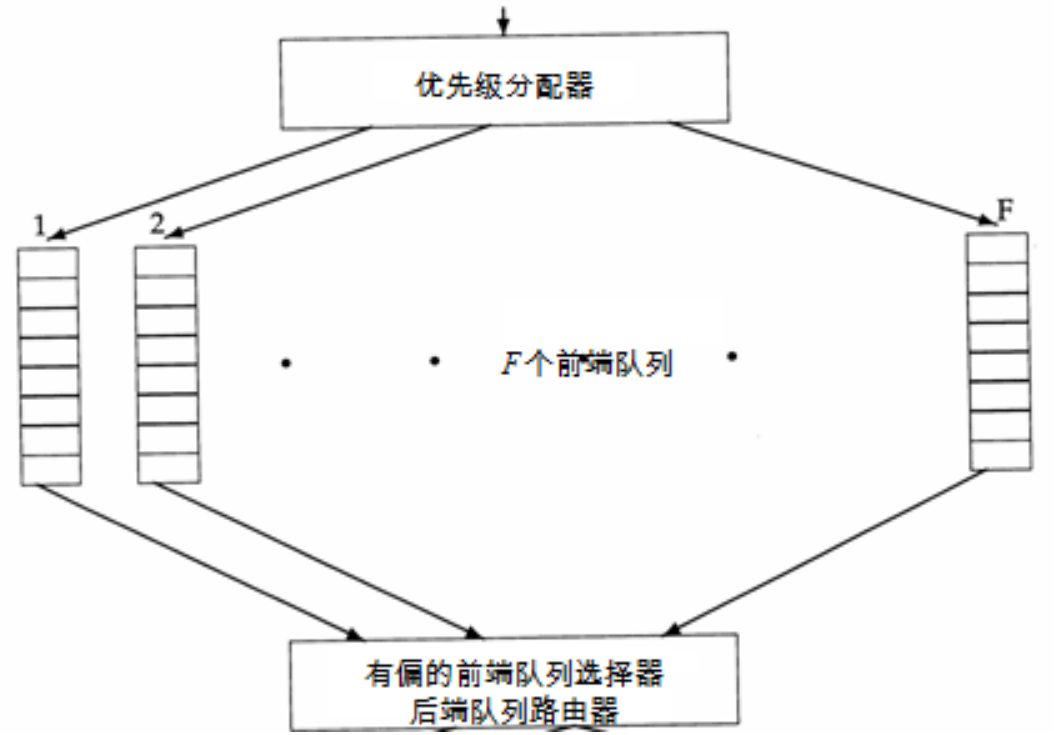
- URL从上部流入缓冲池
- 前端队列(Front queue)
  - 管理优先级
- 后端队列(Back queue)
  - 实现礼貌性
- 每个队列都是FIFO

Mercator 中的待采集URL缓冲池：前端队列(Front queue)



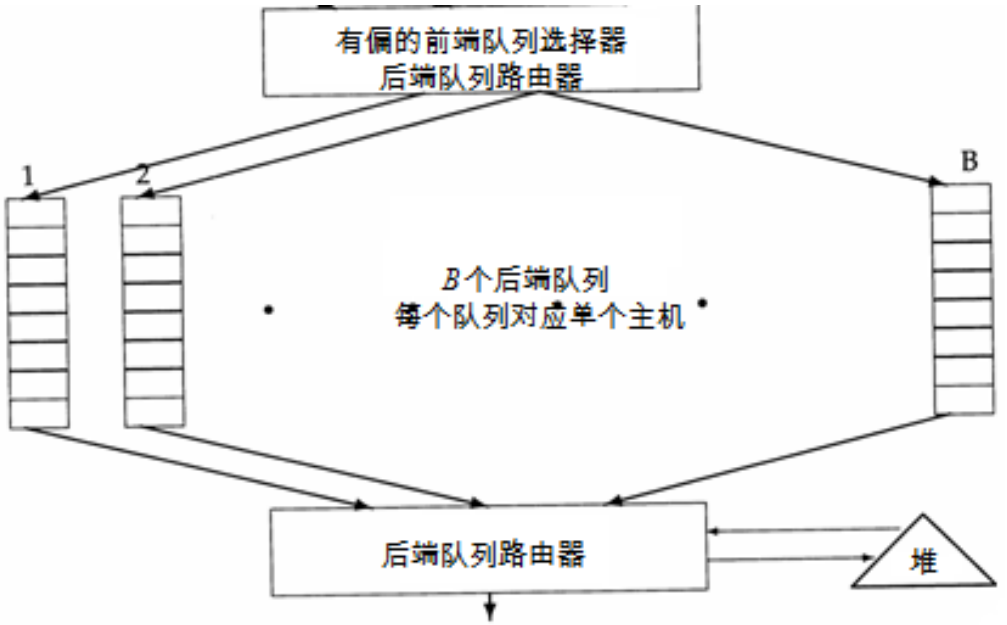
- 优先级分配器给每个URL分配一个0到F之间的优先级整数
- 然后将URL添加到相应的队列中
- 分配优先级可以基于启发式信息：更新率、PageRank等等

Mercator 中的待采集URL缓冲池：前端队列(Front queue)



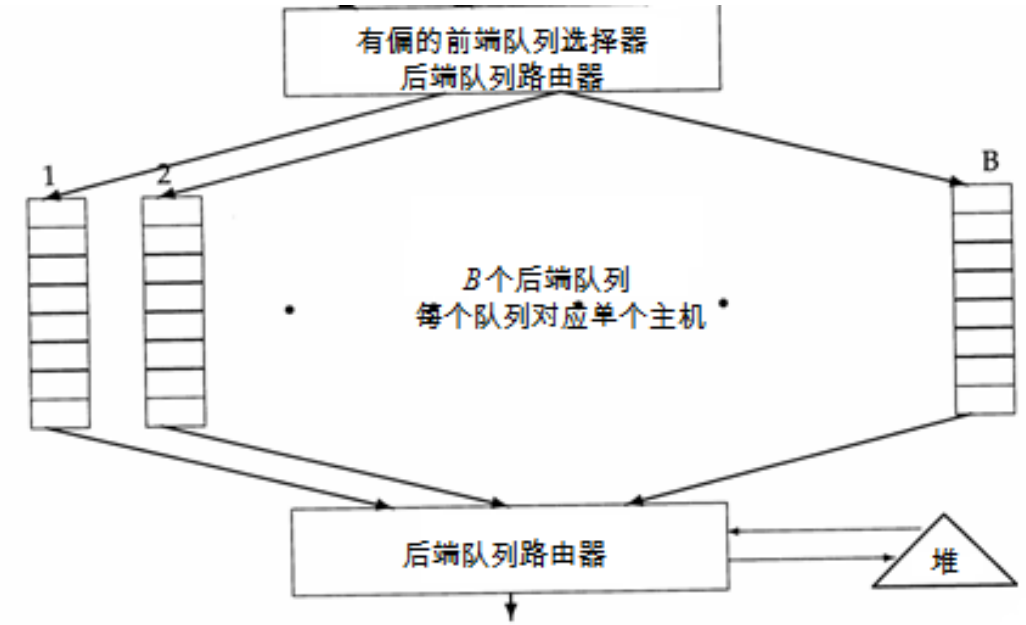
- 从前端队列中进行选择由后端队列发起
- 选择一个前端队列来选择下一个URL：轮询法(Round robin)、随机法或者更复杂的方法
- 但是上述选择过程倾向于高优先级的前端队列

Mercator 中的待采集URL缓冲池：后端队列(Back queue)



- 恒定情形1： 当采集器在运行时，每个后端队列不为空
- 恒定情形2： 每个后端队列中仅存放来自同一主机的URL
- 维护一张主机到后端队列的表

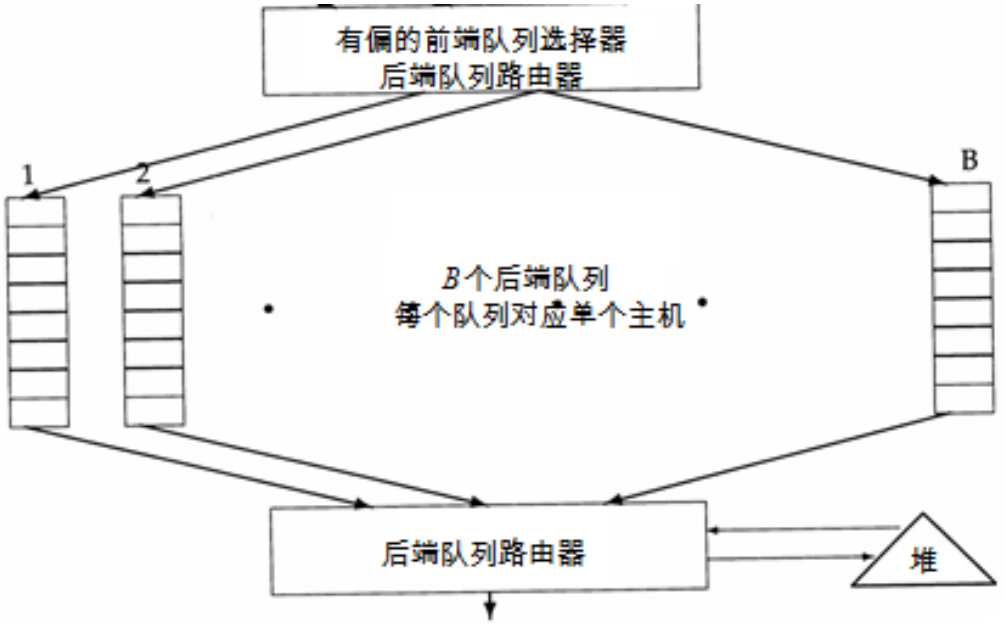
Mercator 中的待采集URL缓冲池：后端队列(Back queue)



在堆中：

- 每个后端队列对应一个元素
- 元素值为该队列对应的主机重新访问的最早时间 t_e
- 该时间 t_e 由下列因素确定
  - (i) 上次访问该主机的时间
  - (ii) 时间间隔的启发式方法

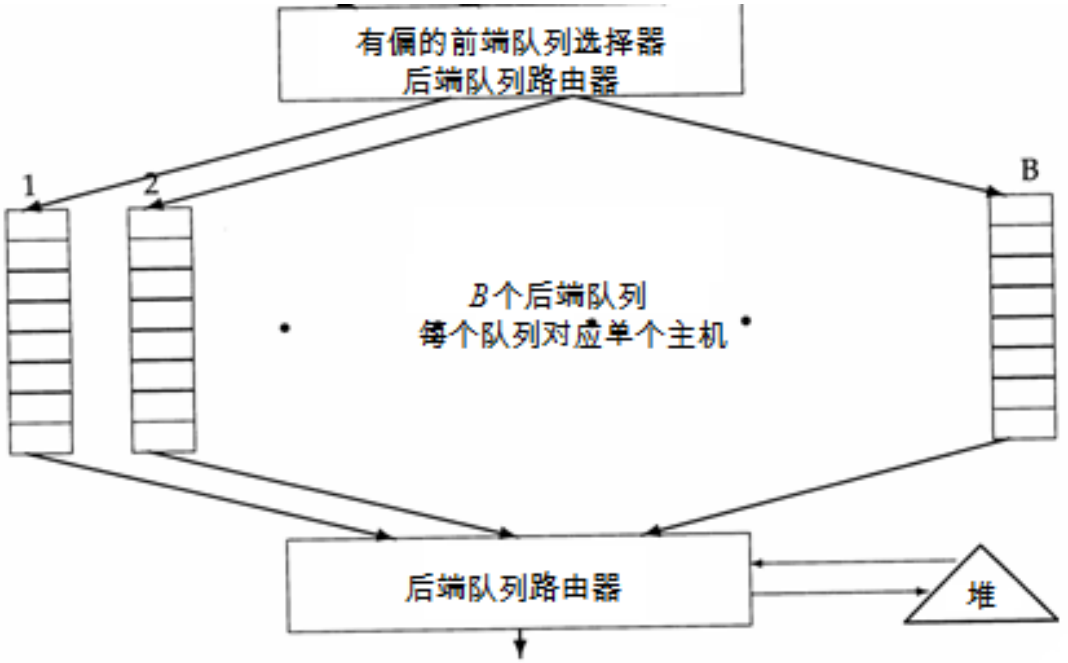
Mercator 中的待采集URL缓冲池：后端队列(Back queue)



抓取器与后端队列交互方法：

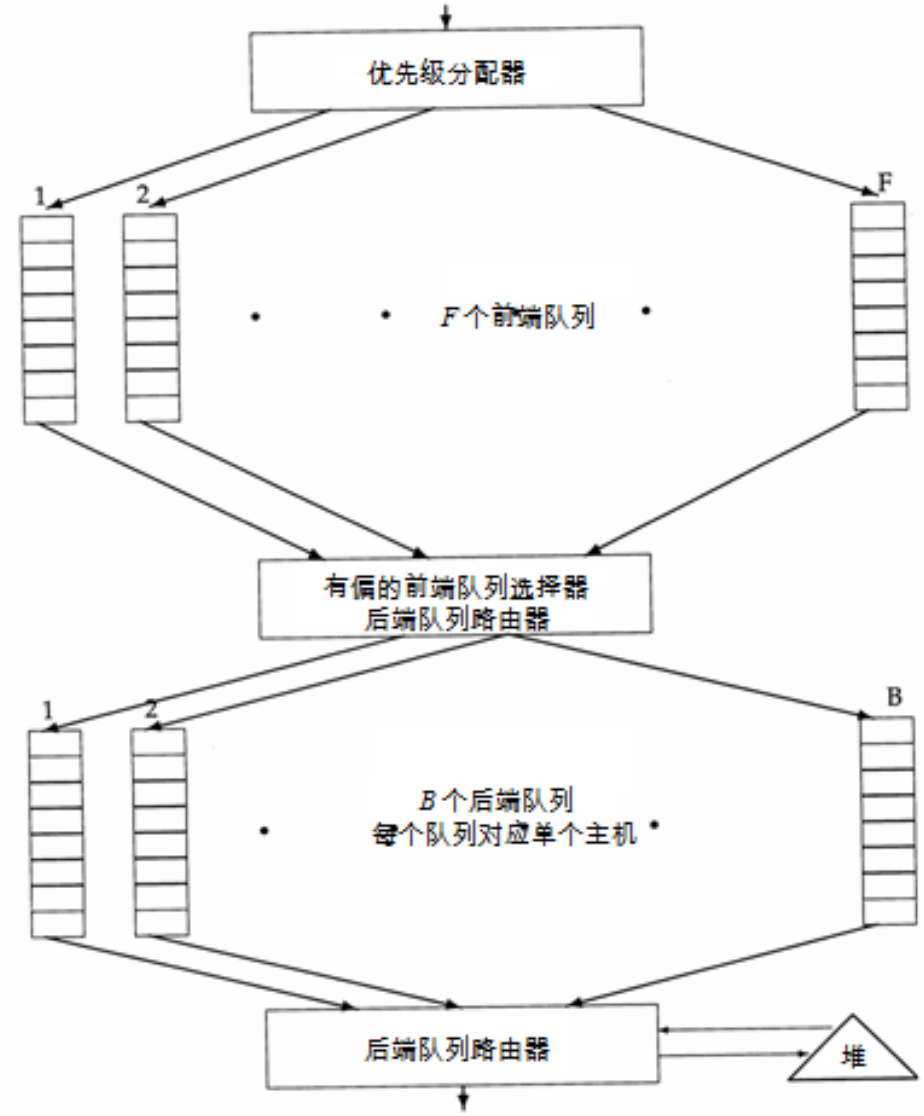
- 重复下列操作：
  - (i) 抽取堆中的当前根节点 q (q 是一个后端队列)
  - (ii) 抓取q中的头部URL u
  - . . .
- . . .直至 q 为空...
- (也就是说一直抓到 u为q中最后一个URL为止)

Mercator 中的待采集URL缓冲池：后端队列(Back queue)



- 一旦后端队列 q为空:
- 重复下列操作
  - (i) 从前端队列中将一系列URL u推入
  - (ii) 将 u 加到相应的后端队列中. . .
- . . . 直到得到一个u，u的主机没有对应的后端队列为止
- 然后将 u 放入 q 并为它建立一个堆

Mercator



- 前端队列管理优先级
  - 前端队列数目、优先级分配、及队列选择策略决定了系统中优先级的性质
- 后端队列保证礼貌性
  - 后端队列的数目控制了在保证礼貌性的同时所有采集线程的忙碌程度

爬虫陷阱(Spider trap)

- 一些服务器可以产生无穷的链接网页序列导致爬虫陷阱
  - 静态网页构成闭环回路
    http://foo.com/bar/foo/bar/foo/bar/foo/bar/.....
  - 在文件系统中的符号链接（symbolic link）可导致爬虫陷阱
  - 类似日历的动态页面可以产生无穷的链接网页序列

参考资料

- 《信息检索导论》第20章

谢　谢！