文章编号: 1001 - 9081 (2006) 01 - 0021 - 04

dPageRank——一种改进的分布式 PageRank算法

陈再良,凌 力,周 强
(复旦大学 通信科学与工程系,上海 200433)
(0151037@fduan. edu. cn)

摘 要:回顾了传统的 PageRank计算方式,分析了等级泄漏和悬挂页面问题的解决方法。介绍了分布式 PageRank的计算原理和评价原则,在分析两种现有分布式 PageRank算法的基础上,提出了一种改进的分布式 PageRank算法,通过实验对该算法的性能进行分析评价。

关键词:搜索引擎; PageRank;分布式
中图分类号: TP393; TP309 **文献标识码**: A

dPageRank — Improved distributed PageRank algorithm

CHEN Zai-liang, LING Li, ZHOU Qiang
(*Department of Communication Science and Engineering, Fudan University, Shanghai* 200433, *China*)

Abstract: The traditional PageRank algorithm was reviewed. The problems of rank-leaking and dangle-pages were discussed. Then the theory and evaluation standard of distributed PageRank algorithm were introduced, and two published experimental algorithms were discussed. After that, a new distributed PageRank algorithm was proposed, and an experiment was set up to analyze the performance of this algorithm. At last, the conclusions were summarized and future research directions were discussed.

Key words: search engine; PageRank; distributed

传统的 PageRank是一种建立在所有网页链接拓扑图上的、集中式的计算技术。随着网页数量的快速增长,集中式的网络搜索引擎已经不能在性能上满足需求。目前的几乎所有搜索引擎均实现了分布式架构,其 Crawler系统 (网页抓取系统)和网页数据库均分布在几百乃至上千台服务器上,例如 Google的服务器就超过了 15 000台[1]。在这种情况下, PageRank只有采用分布式的算法,才能和目前的网络搜索引擎结构相结合,并在性能上满足快速更新的要求。

1 分布式 PageRank算法实现面临的主要问题

PageRank采用分布式计算时,整个网络拓扑图被划分成若干部分,计算时必然造成各个部分之间的链接失效。例如,图 1网页 *A*有指向网页 *E*的链接,但网页 *A*和网页 *E*分别在节点 *N*1和 *N*2中。节点 *N*2在计算网页 *E*的 PageRank时无法从本地获得这条链接的信息。这种情况是导致分布式 PageRank计算产生误差的主要原因。

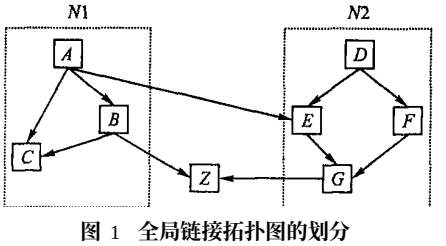


图 1 全局链接拓扑图的划分

因此,各个节点在计算 PageRank时必须进行相互通信,并通过适当的融合算法生成全局 PageRank,从而降低和集中式 PageRank结果的误差。但是,各节点间的通信和融合必将导致计算量的增加和性能的降低。所以,如何在提高精确度的基础上降低通信量,是分布式 PageRank算法主要考虑的问题。

2 分布式 PageRank计算原理

2.1 PageRank的迭代计算

PageRank的基本思想主要是来自传统文献计量学中的文献引文分析[13],即一篇文献被其他文献引用越多,则文献质量就越高。根据这个原理,得到了 PageRank的基本定义:设 *u*为一个网页, *Out*(*u*) 表示从网页 *u*向外的链接数目, *Point*(*u*) 表示链接到网页 *u*的网页集合, *PK*(*u*) 表示网页 *u*的 PageRank值, *C*为规范化因子。网页 *u*的 PageRank计算基本公式[4]如下:

$$PK(u) = C \sum_{v \in Point(u)} PK(v) / Out(v) \qquad (1)$$

式 (1)仅仅是一个 PageRank的定义公式。如果要计算网页集合中所有网页的 PageRank值,必须利用这个公式进行反复迭代。假设 *S*为所有网页的总数,初始化时每个网页的 PageRank都赋以 1/*S*, 然后根据公式 (1)反复迭代计算,直到最后得到的 PageRank值收敛于一个相对固定的数。算法描述如下:

$\forall u \in S: PK(u)_0 = 1/|S|$ //set initial PageRank

while($|PK(u)_i - PK(u)_{i-1}| > \xi$){ //whether convergent or not

for each $u \in S$:

$PK(u)_i = \sum_{v \in Point(u)} PK(v)/Out(v)$ //iterate using equation (1)

}

收稿日期: 2005 - 07 - 15;**修订日期**: 2005 - 09 - 08

作者简介:陈再良 (1982 -),男,江苏无锡人,硕士研究生,主要研究方向:通信与信息系统; 凌力 (1967 -),男,浙江临安人,副教授,硕士,主要研究方向:网络通信、网络安全; 周强 (1973 -),男,山东人,讲师,博士研究生,主要研究方向:计算机网络.

2.2 等级泄漏和悬挂页面的去除

前面所定义的 PageRank计算公式有一个假设前提:所有的网页链接形成一个强连通图。但实际的网络链接拓扑没有这么理想化,会存在大量的没有外出链接的独立网页,这种独立的网页称为悬挂页面 (Dangle Page)[12]。悬挂页面的存在将导致一个主要问题:等级泄漏 (Rank Leak)。等级泄漏将导致所有页面的 PageRank总值不断减小。

针对这个问题,必须引进一个“rank source”(等级源)来不断地补充每个网页的 PageRank值,使 PageRank的分配不完全依赖着链接。修正的 PageRank计算公式[4]为:

$$PK(u) = d \sum_{v \in Point(u)} PK(v)/Out(v) + (1 - d)$$

式中定义一个衰减系数 $d(0 < d < 1)$, d的值通常为 0.85左右。

2.3 分布式 PageRank算法评价方法

评价分布式 PageRank算法性能的主要指标是和集中式 PageRank算法的差异度。参考文献[1]中提出了一种量化计算方法。假定集中式 PageRank计算所得的网页级别向量是 G;而分布式 PageRank计算所得的网页级别向量是 D。如果要计算排列在前 k个网页的差异度,假定 $G(k)$和 $D(k)$是 G和 D前 k个网页的序列,序列 G的前 k个页面集合为 K。那么前 k个页面的误差矩阵定义为:

$$K_{topk}(G,D) = \sum_{\{i,j\} \in K} K_{\{i,j\}}(G,D)$$

其中,如果 $G(k)$同时包含网页 i和 j,但是 $D(k)$中网页 i和 j均不存在,那么 $K_{\{i,j\}}(G,D)$等于 1;如果网页 i和 j在 G和 D中的相对位置不同,则 $K_{\{i,j\}}(G,D)$等于 1;否则, $K_{\{i,j\}}(G,D)$等于 0。

加入网页的相对重要程度这个因素后,本文修正了参考文献[1]的评价公式,前 k个页面的误差值定义为:

$$K_{tipk}Dist(G,D) = \frac{\sum_{\{i,j\} \in K} K_{\{i,j\}}(G,D) * (PK(i) + PK(j))}{k * (k-1)^2 * \sum_{i \in K} PK(i)/2}$$

计算公式中的 $PK(i)$和 $PK(j)$分别表示网页 i和 j在集中式 PageRank算法下得到的 PageRank值。如果序列 G和 D的序列完全相反,则 $KDist(G,D)$就取到最大值 1。若 G和 D序列完全相同,则为 0。

3 改进算法 dPageRank的提出

3.1 改进算法的基本思想和优点

1) 降低与集中式 PageRank的误差

在参考文献[1]中,提出了 ServerRank的算法,它的基本思想是基于不同的 crawler节点抓取不同网站的网页,减少了节点间的链接数,从而降低了与集中式 PageRank的误差。但是该算法给 crawler系统带来了负担,实际分布式搜索引擎很难按照这个模式设计。而且随着网站数量的增长,特别是一些小型网站的出现,使得节点数量变多,增加了不同节点间总通信量。

所以,在改进的分布式 PageRank算法——dPageRank中,并没有直接采用这种方式来降低误差度。而是在实现节点间通信和全局 PageRank融合时,采取了类似 ServerRank的方法,即按照重要性,给每个节点赋予不同的权值。在每个节点根据基本 PageRank公式迭代计算完毕后,根据网页所在节点的权值修正其 PageRank值。

同时,由于节点间通信只能影响直接参与节点间链接的网页。为了把这种影响扩散至节点内部的所有网页,算法在节点间通信后,每个节点继续迭代计算数次,使得节点间通信能更全面的影响网页 PageRank值。

2) 降低局部 PageRank之间的通信量

在参考文献[2]中,提出的基于目标页面分块的算法,这种通信方式具有一定的通用性。但是在分成 N个节点的情况下,该算法至少需要通信 $2\log_2 N$次。这样的通信次数显得过多,需要设法减少。

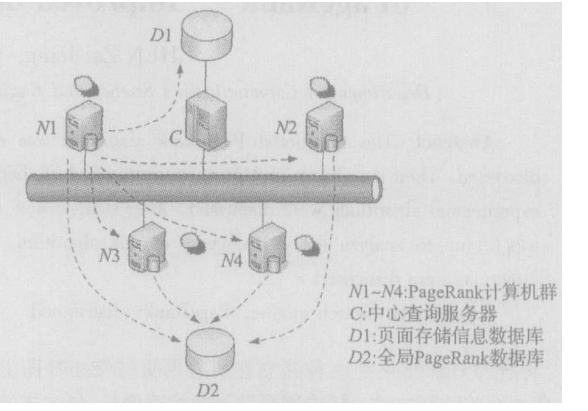


图 2 dPageRank的通信过程

dPageRank采用如图 2的计算结构:首先,在网络上建立一个中心查询节点,存放所有网页和所存放节点的对应关系。其次,在计算局部 PageRank时,先完全舍弃节点间的链接,仅计算存在本地数据库中的所有

网页的 PageRank值。在计算完成之后,每个节点再计算本地为其他节点的网页贡献的 PageRank值,并在节点间相互通信时通过中心查询节点,将贡献的 PageRank值叠加至对应的网页 PageRank上。这种方法完全保留了节点间的链接信息,因此在节点间通信时,能使 PageRank的精确度得到很大提高。同时,通过引入中心查询节点, dPageRank在节点间进行通信时,只需要集中通信 1次或多次。

3.2 关键步骤算法

3.2.1 局部 PageRank计算

在搜索引擎的 crawler系统抓取完一定数量的网页之后,就开始局部 PageRank的计算。首先需要建立出度库和反向链接库,结构如表 1。

在 Outdegree.bdbB库中, key是代表源网页的 Docid, data是源网页的外向链接数 outdegree以及这个网页的 PageRank值。如果该网页是悬挂页面,则 outdegree为 0,所有源网页的 PageRank值均初始化为 1,而悬挂页面的 PageRank值则初始化为 0。 Inverted.bdbB存放了所有的反向链接信息, key是代表目标网页的 Docid, data是源网页的 Docid,以及该链接的重复次数 repeat。

表 1 局部 PageRank数据库

名称	格式	内容	功能
Outdegree.bdbB	Berkeley DB B + tree	S_Docid/ <Outdegree, PageRank >	存放本地所有网页出度和 PageRank值
Invert.bdbB	Berkeley DB B + tree	D_Docid/ < S_Docid, Repeat >	存放本地反向链接信息

局部 PageRank的计算根据 PageRank的基本计算公式而成,其计算公式如下:

$$PK(i) = (1 - d) \sum_{j \in Point(i)} PK(j) \times RPT(j, i) / Out(j) + d$$

式中 $RPT(j, i)$为网页 j指向网页 i的重复链接数, d取 0.15。

dPageRank算法中只去除第一级悬挂页面。首先计算本地所有源页面的 PageRank值,迭代数次后,得到基本稳定的 PageRank向量。然后,利用 PageRank的基本计算公式,进一步计算得到所有悬挂页面的 PageRank值。

计算悬挂页面 PageRank值的流程如下:

遍历 Outdegree. bdbB,取出 outdegree为 0的记录,得到悬挂页面的 Docid。然后从 Inverted. bdbB中查出指向这个悬挂页面的所有源页面以及链接重复数。将所有源页面的 PageRank值除以这些源页面的 Outdegree,并全部相加,便得到了这个悬挂页面的局部 PageRank值。

3.2.2　局部 PageRank相互通信

由于没有去除所有的悬挂页面,每个节点的 PageRank总值在计算过程中将不断减少,必须在节点间的通信之前采用一定的补偿算法。补偿算法分以下两种:

1) 将所有节点的 PageRank总值恢复至初始值

具体算法如下:假设节点 $N1$的 PageRank初始总值为 $P1$,迭代计算后下降为 $P2$。在局部 PageRank进行通信之前,将节点 A中每个源页面的 PageRank值乘以 $P1/P2$,这样就使节点 $N1$的 PageRank总值恢复至 $P1$。

这种算法可以充分消除悬挂页面带来的等级泄漏问题,即使某些节点的悬挂页面很多,也不会因为等级泄漏过多而造成误差。

2) 按比例恢复每个节点的 PageRank总值

具体算法是:假设共有 N个节点,开始计算前这 N个节点的 PageRank总值是 $P1$,迭代计算后所有页面的 PageRank总值降至 $P2$,则补偿时,将所有节点上源页面的 PageRank值乘以 $P1/P2$,这样所有页面的 PageRank总值就会恢复至初始值,但是每个节点上网页得到的补偿却各不相同。

这种算法在某种程度加大了等级泄漏效应。但是,从实验的结果看来,由于悬挂页面较多的节点,在集中式计算时 PageRank总值也会损失较多,所以这种算法得到的结果将更接近集中式 PageRank的计算结果。

在后面的实验中,将比较这两种算法的精确度情况。

局部 PageRank之间的相互通信计算过程如下:

```
For页面 x in节点 N1
  If N1.x outdegree == 0 Then
    查询得到 x是属于节点 N2的源页面
    N2.x PageRank += N1.x PageRank - d
  Endif
Next
```

这样计算的原理是,网页 x在节点 $N1$中的 PageRank值,实际上代表了节点 $N1$中所有网页对网页 x的 PageRank贡献。但是,网页 x是由节点 $N2$抓取,是属于 $N2$的源页面,所以最终计算时应当以节点 $N2$中 x的 PageRank值为准。这就需要将节点 $N1$中 x的 PageRank值叠加至节点 B上。另外,考虑到衰减系数 d的影响,如果一个网页在多个节点上出现,则这个 d值就会叠加多次,因此要在节点间通信时扣除 d值。

3.2.3　全局 PageRank的融合生成

当所有节点的局部 PageRank相互通信之后, PageRank得到增加的只是直接涉及节点间链接的网页,而被这些网页所指向的页面却没有得到增益,因此,为了避免这些网页 PageRank值的损失,可以在局部 PageRank完成一次通信后,在局部节点上做进一步的 PageRank扩散计算。计算方式:

```
For页面 x是节点 N1的源页面
  If N1.x PageRank在通信中获得增长
    查找 Invert bdbB,获得所有被 x所指向的网页
    将 x在通信中获得的 PageRank增加值,除以 x的出度,
    叠加至 x指向的所有网页上
  End if
Next
```

融合生成全局 PageRank的是一个集中式的过程,具体如下:

计算的中心节点要有一个存放所有网页 Docid的数据库,从这个库中取出某个网页 x的 Docid,查询这个网页是属于哪个节点的源页面。如果 x是 $N1$节点的源页面,则 x最终的 PageRank值就是 $N1.x$ $PageRank$,如果 x不属于任何一个节点的源页面,则证明网页 x是一个悬挂页面,需要进一步查询 x在哪些节点上,将 x在所有节点上的 PageRank值相加,并减去 $(n - 1) * d$,其中 n为页面 x所属节点的数目, d是衰减系数。这样就得到了所有悬挂页面的最终 PageRank值。

4　改进算法实例分析

4.1　测试数据和实验平台介绍

测试数据来源于 2005年 1月抓取的一批网页,源页面总数为 4 000个,由源页面解析出的页面总数为 50 797个,链接总数为 118 562,节点总数为 4,每个节点的数据库具体情况如表 2。

表 2　测试数据所含网页数

数据库 ID	源页面数	悬挂页面数	总共链接数 (不含重复链接) / (含重复链接)
1	1 000	11 737	24 151 / 29 329
2	1 000	11 037	14 938 / 22 763
3	1 000	15 043	48 977 / 63 413
4	1 000	10 499	30 496 / 35 266

这 4部分的相互之间链接情况如表 3。

表 3　节点间链接情况

数据库 ID/节点间链接数目	1	2	3	4
1	×	2	NULL	2
2	178	×	NULL	NULL
3	976	NULL	×	NULL
4	2 466	37	NULL	×

根据参考文献 [1]的测试数据,其节点间链接数占链接总数的 3.4%,而在这次的实验数据中,节点间链接数占总链接数的 3.08%,比较符合实际情况。

实验操作系统为 RedhatLinux9.0,编程语言为 C,数据库为 BerkeleyDB 4.2。

4.2　测试结果分析

4.2.1　精确度的分析

在实验中,采用了 4种不同算法,并比较了之间的差异,测试结果见图 3。

图 3(b)是参考文献 [1] 所用的 ServerRank算法测试结果,作为对比参照。图 3(a)表示了采用 dPageRank算法 KDist值随着网页数量的增加的变化状况, 4条曲线表示 4种

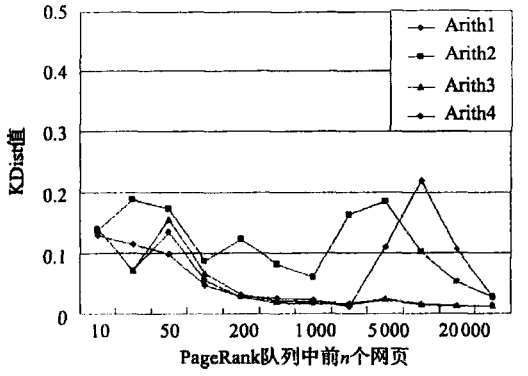
不同的算法:

Arith1:不对 PageRank的损失进行补偿。

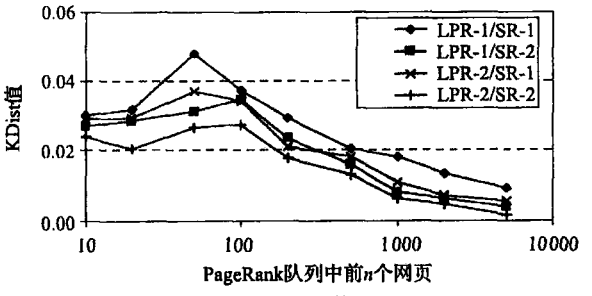
Arith2:采用平均补偿算法,即每个节点 PageRank统一补偿至初始值。

Arith3:采用按比例补偿算法,即每个节点 PageRank值按比例补偿。

Arith4:首先采用 Arith3,在节点之间通信以后,在每个节点采用扩散算法,使局部 PageRank的通信影响更多的网页。



(a) dPageRank算法



(b) ServerRank算法

图 3 精确度的分析

采用平均补偿算法,前 10 000个页面和集中式算法的 KDist值保持在 0.1以上,KDist值始终超过了按比例补偿算法。因此,平均补偿算法并不适用于分布式 PageRank计算。

与平均补偿相对的是,不对 PageRank总值的损失做任何补偿,这种算法从 2000个页面至 10000个页面会出现很大的乱序。其中的原因是:对于 PageRank较小的网页而言,如果不进行补偿,它们和一些悬挂页面的相对位置将发生很大的变化,这就是在 2000至 10000这一段页面 KDist突然升高的原因。

从 KDist值看来,采用按比例补偿是最佳算法,在前 100个页面中,KDist值基本维持在 0.05～0.15之间,在前 100个页面之后,KDist值便逐渐下降至 0.01～0.02之间,即表示每 10个页面中会出现一对乱序,已经达到了和 ServerRank算法相近的精确度,是相当理想的结果。

Arith3和 Arith4的结果非常接近,Arith4的 KDist值在前 1000个页面范围内稍低 0.005～0.01。这是由于 Arith4在局部 PageRank通信后,每个节点采用了 1级扩散算法,使得 KDist值有细微的改善。由于 1级扩散算法是每个节点分布式计算,且计算量较小,在实际计算中可以考虑适当采用这种算法。

4.2.2 局部之间相互通信量分析

在上面的实验数据中,节点间互相通信的数据量由节点间链接紧密程度来决定,表 4是 4个节点之间的链接页面数。

表 4 节点间链接页面数

数据库 ID/链接涉及页面数目	1	2	3	4
1	×	2	NULL	2
2	34	×	NULL	NULL
3	1	NULL	×	NULL
4	145	37	NULL	×

平均每两个节点之间的链接页面数是 18.4,占每个节点源页面总数的 1.84%。每一对链接传送的链接内容为 <Docid(long 32位),PageRank(float 32位)>,故共需 64位(8个字节)。平均每两个节点之间的通信量为 18×8B=144B。

这个数据仅仅是在每个节点存储 1000个页面下测得的,在真实环境下,每个节点存储的页面数应当在千万级以上,即页面数量要增加 104。这样,节点间链接数目应当会成 108增长。但是,由于 dPageRank的算法只考虑参与链接的页面总数,因此通信量的增长应当与页面数增长速度保持一致,即 104。这样,每两个节点之间的通信量,在 1MB～10MB之间。

将这个和数据参照文献[1]的实验结果进行对比:ServerRank算法中,平均每两个节点要传递 10条消息,每条消息大小为 2.1kB。因此,在采用 ServerRank的情况下,平均每两个节点之间的通信量为几十 kB。

虽然这个数据仅是 dPageRank的节点间通信量的 1/100,但是,由于 ServerRank算法是基于将网页按网站服务器进行划分的,因此,在实验网页数为 1049271的数据量下,该算法划分了 285个节点,平均每个节点网页数为 3681个。在这种情况下,它需要的总通信量为 285×(285-1)/2×10k=400MB。而如果利用 dPageRank计算方式,可以灵活的分成几十个节点,以 20个节点为例,每个节点的网页数量为 5万左右。这样,根据上面提出的公式,平均每两个节点之间的网页链接数为 50000×1.8%=900个,通信的信息量为 900×8B=7.2kB。总的通信量为 20×7.2kB=140kB,远小于 ServerRank算法的通信量,显示了改进算法性能的提高。

5 结语

dPageRank算法是 PageRank算法在分布式搜索引擎中的具体应用。一方面,该算法采用保留节点间链接、节点权值补偿、节点内部扩散等方法,保证了结果的精确度;另一方面,在深入分析悬挂页面和等级泄漏问题的基础上,算法仅去除一级悬挂页面,有效的减少了节点之间的通信量。本文还通过实验实现了该算法,并和文献[1]中已有算法成果的对比,说明 dPageRank算法在保证精确度的同时,降低了分布式节点之间的通信量,取得了理想的效果。

dPageRank算法虽然已在实验环境中表现出良好的性能,但是,在投入实际应用前,该算法还需要作进一步测试和完善。在以后的工作中,将就以下两方面问题作进一步研究:

1)和词频位置加权算法的结合,设计出一种较好的结合算法,使搜索引擎能返回和用户查询内容最匹配的结果。

2)对 PageRank算法的改进:通过引入一些动态因素(如用户的点击情况),来修正 PageRank的计算结果,从而获得更符合用户需求的网页排名。

参考文献:

[1] WANG Y, DEWITT DJ. Computing PageRank in a Distributed Internet Search System[A]. Proceedings of the 30th VLDB Conference[C]. Toronto, Canada, 2004.

(下转第 36页)

事务层是 SIP协议栈的核心组成部分,负责 SIP协议相关状态机的处理。用户代理维护不同状态机类型的链表,对于由消息层收到的协议包和来自对话层的发出协议数据包的请求,事务层都会为其创建一个事务状态机并将其加入相应的链表。状态机随着系统的运行不停的运转,根据状态机目前的状态和相关的输入输出事件以及时间事件,状态机进行变迁并完成相关的动作。事务层以回调函数的形式通知对话层状态机的变迁和事件的产生,应用程序可以对感兴趣的事件进行处理。

对话层直接响应应用程序的请求,申请创建或撤销与另一个 SIP用户间的对话,对话层使用下层事务层的服务为对话创建对应的客户或服务端状态机。状态机的运行结果以事件的形式通知给上层用户,会话层响应下层的事件通知并将用户感兴趣的信号或错误直接提交给用户应用程序。终端控制模块作为协议栈的应用程序和对话层互相调用对方提供的接口函数完成信令的转换。在信令转换过程中代表远端的逻辑终端和用户代理中的对话构成互相引用的一一对应关系。

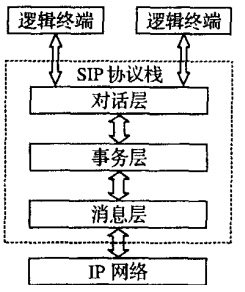


图 4 SIP模块中的 SIP协议栈

4 系统的扩展

目前 IP网络上流行的信令协议除了 SIP之外,还有诸如 H.323[4],MGCP[5],Megaco[6]等,由于本系统的主要呼叫控制交换功能都集中于终端控制模块,通过对系统的扩展,只用增加相应的协议转换模块就能够使网关能够接入支持这些协议的其他设备。对于同样使用 SDP协议作为媒体描述的协议(SIP,Megaco),终端的媒体描述信息可直接传递给相应的信令模块,而对于像 H.323一样使用其自有的媒体协商协议(H.245)则需要做一定的转换。

5 结语

本文提出了一种可扩展多协议的多状态机媒体终端交换机制。通过增加此核心的媒体终端交换模块,使各个模块之间的多个状态机通过这个核心模块周转,系统结构清晰。该系统高度模块化,具有较强可移植性、可维护性。从根本上解决 MoIP网关对于各种多媒体终端设备,多协议的接入问题,并提出了基于 SIP协议的信令接入的完整解决方案。

参考文献:

[1] ROSENBERG J, SCHULZRINNE H, CAMARILLO G, *et al*. SIP: Session Initiation Protocol[S]. IETF RFC 3261, June 2002. http://www.ietf.org/rfc/rfc3261.txt

[2] HANDLEY M, JACOBSON V. SDP: Session Description Protocol[S]. IETF RFC 2327, 1998.

[3] ROSENBERG J, SCHULZRINNE H. An Offer/Answer Model with Session Description Protocol(SDP)[S]. IETF RFC 3264, 2002.

[4] JONES P E. Draft H.323v4: Paket Based Multimedia Communication Systems[S]. ITU-T Recommendation H.323, 2000.

[5] GREENE N, RAMALHO M, ROSEN B. Media Gateway control protocol architecture and requirements[S]. IETF RFC2805, 1999.

[6] CUERVO F, GREENE N, RAYHAN A, *et al*. Megaco Protocol Version 1.0[S]. IETF RFC3264, 2000.

(上接第 24页)

[2] MANASKASEMSAK B, RUNGSAWANG A. Parallel PageRank computation on a gigabit PC cluster[A]. Proceedings of the 18th International Conference on Advanced Information Networking and Application[C], 2004.

[3] SANKARALINGAM K, SETHUMADHAVAN S, JAMES C. Browne. Distributed PageRank for P2P systems - High Performance Distributed Computing[A]. Proceedings of the 12th IEEE International Symposium[C], 2003.

[4] BRIN S, PAGE L. The Anatomy of a Large-Scale Hypertextual Web Search Engine[A]. Proceedings of the 7th International World Wide Web Conference (WWW7)[C], 1998.

[5] HAVELIWALA TH. Efficient Computation of PageRank[A]. Stanford University Technical Report[C], 1999.

[6] YAMAMOTO A, ASAHARA D, ITAO T, *et al*. Distributed PageRank: A Distributed Reputation Model for Open Peer-to-Peer Networks[A]. International Symposium[C], 2004.

[7] KAMVAR S, HAVELIWALA T, GOLUB G. Adaptive Methods for the Computation of PageRank[EB/OL]. citeseer.ist.psu.edu/kamvar03adaptive.html, 2003.

[8] KAO B, LEE J, NG C Y, *et al*. Anchor Point Indexing in Web Document Retrieval[A]. IEEE transaction[C], 2000.

[9] CAN F, NURAY R, SEVDIK A B. Automatic performance evaluation of Web search engines[D]. Department of Computer Engineering, 2003.

[10] CARAMIAA M, FELICIB G, PEZZOLICA. Improving search results with data mining in a thematic search engine[A]. Computers & Operations Research[C], 2004.

[11] The Black Box Group. PageRank Explained[EB/OL]. www.rankwrite.com, 2002.

[12] HAJME BABA. Google的秘密——PageRank彻底解说[EB/OL]. linux.dalouis.com/PageRank_cn.htm, 2002.

[13] 曹军. Google的 PageRank技术剖析[J]. 情报杂志, 2002, (10).

[14] 许涛, 吴淑燕. Google搜索引擎及其技术简介[J]. 现代图书情报技术, 2003.

[15] 常璐, 夏祖奇. 搜索引擎的几种常用排序算法[J]. 图书情报工作, 2003, (6).

[16] 赵新慧, 朱伟. 分布协作式搜索引擎系统的初步探索[J]. 抚顺石油学院学报, 2003, (4).

[17] 刘悦, 杨志峰, 程学旗, 等. 利用链接分析技术提高搜索引擎查找质量的研究[J]. 微电子学与计算机, 2002, (5).

[18] 风元杰, 刘正春, 王坚毅. 搜索引擎主要性能评价指标体系研究[J]. 情报学报, 2004, (1).

[19] 宋睿华, 马少平, 陈刚, 等. 一种提高中文搜索引擎检索质量的 HTML解析方法[J]. 情报学报, 2003, (4).

[20] 阎放, 张海涛, 朱宏谊. GOOGLE搜索引擎 PageRank技术的优化[J]. 情报科学, 2002, 20(12).

[21] 夏祖奇, 黄水清, 赵展春. 基于分类目录的元搜索引擎模型的提出与实现[J]. 情报学报, 2003, (1.)

[22] 冯英健. 第三代搜索引擎[EB/OL]. www.marketingman.net, 2005.