基于P2P的数据索引与查询

姚佳丽 张坤龙 王 珊
(中国人民大学信息学院 北京100872)

摘 要 现有的基于DHT的P2P系统只能通过精确匹配整个数据识别器来查询数据。但用户一般只有部分信息可以确认这些信息,为了在用户需求和基于DHT的P2P系统能力间架起一座桥梁,本文提出了一种新的索引和查询数据的方法。这种方法在数据的XML描述上建立了DHT索引,并方便了Xpath表达式的复杂查询。

关键词 DHT,P2P数据管理,数据索引,查询过程

P2P Data Indexing and Querying

YAO Jia-Li ZHANG Kun-Long WANG Shan
(College of Information,Renmin University of China,Bejing100872)

Abstract Current DHT-based P2P systems only support querying data by exactly matching the whole data identifier. But the users often have only partial information for identifying these data. To bridge the gap between the users' requirement and the DHT-based P2P systems' ability,this paper describes a new approach to indexing and querying data. This approach builds DHT index on top of the data's XML description and complex query facilities on top of XPath expression.

Keywords DHT,P2P data management,Data indexing,Query processing

1 引言

数据查询是P2P系统研究的根本问题:如何在一个大规模的P2P系统中,快速准确地找到满足给定条件的数据?对于这个问题的解决,有两个重要的评价标准,即解决方案的查询能力和系统的可扩展性。

典型的P2P系统,如Napster和Gnutella,主要用于文件共享。在查询的时候,这些系统基于文件名进行字符串匹配,因此查询能力很弱,只能找到文件名中包含或不包含某些字符的文件。在可扩展性方面,Napster使用一个服务器来索引所有的文件,Gnutella使用广播的方式将文件搜索请求发给尽可能多的节点,前者的集中式的设计容易形成服务器瓶颈,后者则对网络带宽的要求较高,两者的可扩展性都不好。

后来出现的P2P系统,如CAN[1]、Chord[2]、Pastry[3]和Tapestry[4],都是基于分布式哈希表(DHT,Distributed Hash Table)的,具有较好的可扩展性。在这些系统中,数据项不再局限于文件,将每个数据项映射到一个码(key),系统根据这个码来决定将数据项存储到哪一个网络节点,并保证从系统中任一节点出发查找任意的数据项需要经历的节点数有一个上限(在CAN中是N^d,在其他几个为$\log N$,其中N为系统中节点总数,d是一个较小的值)。但是这些P2P系统在查询时还是只能对码值进行精确匹配,因此它们的查询能力仍然很弱,在一定程度上甚至还不如原有的P2P系统。

已经有一些研究开始致力于提高这些基于DHT的P2P系统的查询能力,如文[5,6]。文[5]的目标是为P2P系统提供关系代数运算能力。例如通过分解子串来进行匹配的方法是,先将字符串分解成长为n的等长子串,然后以这些子串为码将文件ID存储到DHT中。在查找的时候,也是先将查询字符串做同样的分解,然后使用分解得到的每一个子串查找匹配的文件ID,并按文件ID对所有的返回结果分组,显然满足查询条件的文件ID其分组中的成员个数等于查询字符串分解得到的子串的个数。文[5]指出,从关系代数实现的角度来看,上述过程包含索引存取操作、分组操作和选择操作。文[5]还指出,对P2P系统中复杂查询的处理还需要做更多的研究。

为了增强基于DHT的P2P系统的查询能力,本文在上述工作的基础上提出了一种新的方法来对P2P系统中的数据进行索引和查询。本文第2节给出本文设计的基于DHT的P2P系统的总体结构;第3节说明如何对数据进行描述;第4节提出了一种层次性的索引方法;对数据的查询方法在第5节讨论;最后总结全文。

2 系统总体结构

虽然基于DHT的几个P2P系统在实现细节上差别很大,但是不管是直接还是间接,它们都具有两个DHT函数接口:*put*(*key*,*value*)和*get*(*key*)。举例来说,在一个基于DHT的P2P文件共享系统当中,如果网络节点A要提供一个共享文件,它就以被共享的文件的文件名为参数*key*、以节点A的IP地址为参数*value*调用*put*(*key*,*value*)函数。函数*put*(*key*,*value*)以适当的方式在系统中找到一个网络节点B来存储DHT索引项〈*key*,*value*〉。当网络节点C要获得节点A共享的文件时,它就以要获取的共享文件的文件名为参数*key*调用*get*(*key*)函数。函数*get*(*key*)将找到节点B并将*value*返回给节点C,节点C随后可以直接到节点A上去读

姚佳丽 硕士生,研究方向:P2P环境下的数据管理;**张坤龙** 博士生,研究方向:P2P环境下的数据管理;**王 珊** 教授,博导,研究方向:数据库与知识工程。

取文件。

对于这两个 DHT 函数接口，首先要注意到对 *key* 和 *value* 没有什么限定。虽然在文件共享的例子中 *key* 是文件名，*value* 是 IP 地址，但是这并不是唯一的选择。例如，可以以文件的创建者、创建时间、文件大小等等属性信息一起作为 *key*，也可以将文件本身作为 *value*。在以文件本身作为 *value* 的情况下，文件将被存储到存储 DHT 索引项〈*key*，*value*〉的节点上。其次需要注意的是，根据 *key* 来查找对应的 *value* 的过程是对 *key* 的精确匹配，即在查找时给出的 *key* 必须和索引时使用的 *key* 等同，这也就使得在原有的 P2P 系统中只能按照文件全名而不能按照文件名的一部分进行查找。

只能进行精确匹配是一个很大的局限，因为在多数情况下用户可能只知道关于要查找的对象的部分信息。例如在一个共享 MP3歌曲的 P2P 系统中，用户想按演唱者而不是 MP3歌曲文件名来查找歌曲。又比如在一个 P2P 购书系统中，用户可以给出有关书籍的部分信息：如书名、作者、出版社、书号等等来查询符合条件的书籍。为了提高基于 DHT 的 P2P 系统的查询能力，基于以上的讨论本文提出了一个系统模型，它的网络节点具有图1所示的结构。

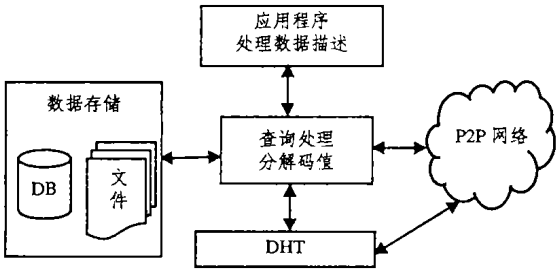

图1　一个网络节点的结构

在图1所示的结构当中，最上层的应用程序处理包括插入和查询数据的请求。在插入数据时，需要给出数据的描述。数据描述的给出方式可以是由应用程序以显式的方式给出，也可以是由系统自动生成。有关数据描述的进一步的内容见本文第3节。对数据的查询，是基于数据描述来进行的，在查询时应用程序可以给出数据的完整描述，也可以只给出数据的部分描述。

图1的中间一层负责数据管理，包括数据存储和查询处理。存储数据时并不局限于文件形式，可以根据数据的特点选择最佳的存储方式，如用关系数据库来存储规范的表格数据。插入数据时，查询处理器一方面以适当的方式将数据在本地存储，另一方还将依据数据描述生成多个码，并使用这些码来建立 DHT 索引，有关如何根据数据描述建立 DHT 索引的详细内容见本文第4节。查询数据时，查询请求中的数据描述以和插入时的数据描述一样的方式被转化为多个码，查询处理器用这些码查找到对应的值，并对找到的值进行处理以便返回最终的查询结果，有关查询处理的详细过程见本文第5节。

在图1中最下层的是 DHT，它的主要功能是构造一个 P2P 网络，提供的 API 中包含前面提到的 *put*(*key*，*value*)和 *get*(*key*)。在进行数据查询时，对于一个给定的码，DHT 层负责找到存储它的网络节点，并返回对应的值。如果这个值是一个 IP 地址，查询处理器就可以通过直接连接和该地址对应的网络节点来完成应用程序的查询请求，否则根据结果进一步发送查询，直到获取目标 IP 地址，或是确认并不存在于网络中。

3　数据的描述

在 P2P 文件共享系统中，文件查找是基于文件名进行的，可以把文件名看作是对文件的简单描述。将这个观点进一步拓展，可以对所有要共享的数据都建立对应的 XML 数据描述，对数据的查询就在这些数据描述的基础上进行。例如，对于一本书的描述如图2所示。

```
〈书〉
  〈书名〉四世同堂〈/书名〉
  〈作者〉老舍〈/作者〉
  〈出版社〉人民文学出版社〈/出版社〉
  〈书号〉ISBN7-02-003263-X〈/书号〉
〈/书〉
```

图2　数据描述示例

这个 XML 数据描述也可以转换成图3中的树，树中一个分支结点对应一个 XML 元素标记，一个叶子结点对应一个 XML 元素的字符数据。

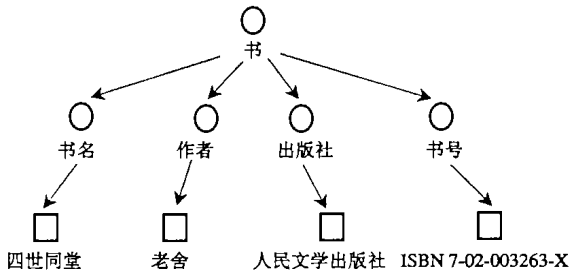

图3　数据描述的树结构图

可以以多种方式来建立 XML 数据描述。对于结构化的数可以根据数据字典信息自动生成其数据描述据，如关系数据库中的一个关系元组。对于半结构化已经用 XML 表示的数据，可以直接使用其 XML 表示作为数据描述。对于有一定结构的数据，如学术论文，可以先自动生成一个 XML 数据描述然后做人工校正。最后，对于无结构数据，可以以人工方式建立其 XML 数据描述。需要说明的是，各个对象的 XML 描述并不一定要具有同样的格式、相同的层次，也就是说对于整个 P2P 系统而言，没有统一的全局约束模式来限定数据描述。各个结点甚至每个结点上的不同对象可以具有不同的层次，而且对于同一对象，也可以增加它的 XML 数据描述类别。这样的处理方式增加了数据描述的灵活性和拓展性，但是由于存在多种数据描述的格式，发出查询的结点至少需要了解待查询数据的其中一种数据描述格式，这样会给客户端查询处理带来一定的复杂度。因此在一般情况下，对于同样类型的对象，往往使用默认的模式描述。如上例所示，属于书的对象一般采用书名、作者、出版社的描述。此外，客户端程序对于不同对象的还可以提供可选的描述，从而保证一致的处理查询和插入操作。

为数据建立 XML 数据描述的优点在于：1）对数据可以做更多的描述。在 P2P 文件共享系统中，文件名的描述能力是有限的，而用 XML 则可以自然地给出更多的有关文件的信息。2）扩大了共享数据的范围。共享的数据既可以大到包括整个文件，又可以小到只涉及一个关系数据库中的元组。由于已将数据与数据描述区分开，使用者甚至可以对数据以外的内容进行描述，例如可以描述一个共享的硬件资源。3）支持更多的查询方式。在 P2P 文件共享系统中，只能按文件名做精确匹配查询。而在本文设计的系统中，可以根据已知的部分

XML 数据描述进行查询。但是在进行查询前，先要根据数据描述建立 DHT 索引。下一节就描述如何建立数据的 DHT 索引。

4 数据的索引

给出对数据的描述之后，最简单直接的方法就是使用整个描述作为码来建立 DHT 索引项。也就是说，调用 DHT 层的 *put*(*key*,*value*)函数时参数 *key* 是整个的 XML 描述。这种方法虽然简单直接，但是查询能力与原来基于文件名的查询一样，只能做整个数据描述的精确匹配。

观察图3中数据描述的树结构，可以提出一个直观的索引方法，即对每个树的叶子结点建立一个 DHT 索引项。这样对于图3中的树就可以建立四个 DHT 索引项，它们的码分别是书名“四世同堂”、作者“老舍”、出版社“人民文学出版社”和书号“ISBN 7-02-003263-X”。建立索引后，可以使用对应于一个叶子结点的部分数据描述而不是整个的数据描述来进行查询，例如可以根据书名“四世同堂”查找该书的其他信息。由于一本书的书号是唯一的，因此使用书号“ISBN 7-02-003263-X”查询时，查询结果也是唯一的，而由于一个作者通常有多本著作，所以使用作者“老舍”查询时，查询结果通常也会是多个。

根据叶子结点建立索引虽然支持使用部分数据描述进行查询，但是它有两个显而易见的缺点。第一个缺点是，它支持的查询是在一个叶子结点的基础上进行的，但有的查询给出的已知条件包含多个叶子结点，例如给出的已知条件包含书的作者“老舍”和出版社“人民文学出版社”。本文第5节中论述了对这种查询的处理方法。第二个缺点是，查询必须知道精确的叶子结点对应的数据描述，例如在上面的例子中的书可以根据书名“四世同堂”找到但是不能根据书名“四世”找到。为了解决第二个问题，需要对上述的索引方法做出改进。在给出改进的索引方法之前，下面先定义什么是一个树结点对应的查询。

q1=/书/书名　　q5=/书/书名/四世同堂
q2=/书/作者　　q6=/书/作者/老舍
q3=/书/出版社　q7=/书/出版社/人民文学出版社
q4=/书/书号　　q8=/书/书号/ISBN 7-02-003263-X

图4　树结点和查询

XPath 是由 W3C 提出的一种 XML 查询语言。将一个 XML 文档看作一棵树，每个查询都是一个 XPath 表达式，包含由多个用“/”分隔的定位步骤，返回的结果是这棵树上的结点集合。例如，对于图3中的树，XPath 表达式“/书/书名/四世同堂”返回的结点集只包含树中最下层最左边一个结点。对 XPath 表达式的形式加以限制，就可以使得树上的一个结点和唯一的一个 XPath 表达式相对应，这个 XPath 表达式以“/”开始，包含从树的根结点到该结点的路径上的所有结点对应的字符串，称作该结点对应的查询。例如，图3中最下层最右边的结点对应的查询是“/书/书号/ ISBN 7-02-003263-X”。图4中的 *q*1～*q*8给出了图3中的叶子结点和叶子结点的父结点对应的所有查询。

给定两个查询 *q* 和 *q*'，如果查询 *q* 的返回结果全部包含在查询 *q*' 的返回结果当中，那么称查询 *q*' 包含查询 *q*，且查询 *q*' 是查询 *q* 的父查询。按照这个定义，叶子结点的父结点对应的查询包含叶子结点对应的查询，例如图4中 *q*1包含 *q*5，且 *q*1是 *q*5的父查询。从一个叶子结点对应的查询获得其父查询的方法很简单，去掉查询表达式的最后一个定位步骤即可，例如去掉 *q*5的最后一个定位步骤“/四世同堂”就可以得到其父查询 *q*1。

第3节已经指出，一个 XML 数据描述可以被转换成一棵树，这样根据 XML 数据描述建立 DHT 索引的方法就是：对于这棵树上的每个叶子结点，先置参数 *key* 为该叶子结点对应的查询、置参数 *value* 为数据所在的网络节点的 IP 地址调用 *put*(*key*,*value*)函数为该叶子结点建立一个 DHT 索引项，然后置参数 *key* 为该叶子结点对应的查询的父查询、置参数 *value* 为该叶子结点对应的查询调用 *put*(*key*,*value*)函数为该叶子结点的父结点建立一个 DHT 索引项。显然如果一个 XML 数据描述的树结构中有 *a* 个叶子结点，则建立的 DHT 索引项共有2*a* 个。例如，图3中的树有4个叶子结点，建立的 DHT 索引项就共有8个，分别是〈*q*5,IP 地址〉、〈*q*6,IP 地址〉、〈*q*7,IP 地址〉、〈*q*8,IP 地址〉、〈*q*1,*q*5〉、〈*q*2,*q*6〉、〈*q*3,*q*7〉、〈*q*4,*q*8〉。从这个例子还可以看到，在这种索引方法中，为叶子结点建立的索引项可以看作是对数据的直接索引，而为其父结点建立的索引项则可以看作是间接索引。

5 数据的查询

对数据的查询是在数据描述的基础上进行的。由于使用 XML 对数据进行描述，因此在查询时候可以使用 XPath 查询语言。与第4节中一样，在查询时对 XPath 表达式也有限制，要求其形式也类似于 UNIX 系统中的文件路径名格式，但是最后一个定位步骤可以包含谓词。例如，“/书/书名/四世同堂”和“/书/书名/四世”都是正确的 XPath 查询表达式，两个表达式的最后一个定位步骤相当于一个包含谓词，这个谓词要求书名包含指定的字符串。

对于给定的一个查询表达式，查询的方法是先直接以该表达式作为参数 *key* 调用 *get*(*key*)函数做精确匹配查询，如果这一步没有结果返回，再以该查询表达式的父查询表达式作为参数 *key* 调用 *get*(*key*)函数，然后在返回的结果中对查询表达式进行匹配，找到匹配的查询表达式后第三次调用 *get*(*key*)函数即可获得最终的查询结果。例如，对于查询表达式“/书/书名/四世”，第一次函数调用 get(“/书/书名/四世”)没有返回结果，第二次函数调用 get(“/书/书名”)将返回“/书/书名/四世同堂”，第三次函数调用 get(“/书/书名/四世同堂”)就会返回存放数据的网络节点的 IP 地址。

可以通过逻辑运算在简单查询表达式的基础上形成复杂的查询。对于单个查询表达式的逻辑非运算，可将其转换为另一个查询表达式来求值，即对原来的查询表达式中最后一个定位步骤所包含的谓词进行否定。对于两个查询表达式的逻辑或运算，查询结果是各个查询表达式的返回结果的并集。对于两个查询表达式的逻辑与运算，可以先求出一个查询表达式的返回结果，然后将未完成的查询交给返回结果对应的网络节点继续求值。

对于通过逻辑运算形成的复杂查询的执行，可能有多种执行方案，这时需要选择执行最优的方案。例如要查找由人民出版社出版的作家老舍写的作品，这个查询是查询表达式“/书/作者/老舍”和查询表达式“/书/出版社/人民文学出版社”的逻辑与运算。一个执行方案是先执行查询表达式“/书/作者/老舍”获得一些存有老舍作品的网络节点的 IP 地址，然后将剩下的查询表达式“/书/出版社/人民文学出版社”发给这些网络节点求值，这些网络节点由于存有完整的数据描述，因此只需要在本地完成剩下的查询并返回结果。另一个执行方案是以类似的方式先执行查询表达式“/书/出版社/人民文学出版社”然后再执行查询表达式“/书/作者/老舍”。一种优化的方法就是在插入数据进行数据描述时给每一项描述增加一个

评判参数，来表示该描述的优先级别。在客户端程序中增加rank授予级别函数，根据该对象所属的类别和重要程度，来授予并存储它的优先级别。对在第3节中提出的插入过程进行拓展，同样该过程可以是系统自动实现的，也可以是通过手工进行修订。在查询过程中，首先分析该查询，计算不同查询方案的优先级，选择代价最小的方案来执行。比如在上例中，考虑到作家的作品数量少于出版社出版的书籍的数量，插入该对象时，"出版社"的描述就要比"作家"的描述具有较低的优先级，从而使得前一种查询执行方案将比后一种查询执行方案好，在查询执行时应该选择前一种方案。

用户在进行数据查询时，以类似于QBE查询语言的方式给出查询请求，这个查询请求随后被转换成多个XPath查询表达式求值。

总结 现有的基于DHT的P2P系统在查询时只能对码进行精确匹配，因而它们的查询能力有限。为了弥补这一缺陷，本文提出了一个系统模型。该系统模型在应用程序和DHT之间加入一个数据管理层，由它负责以适当的方式存储数据，并根据数据描述建立DHT索引和处理查询请求。

将数据和对数据的描述区分开，极大地扩展了系统能够处理的数据类型。数据描述本身是一个XML文档，可以将这个XML文档转换成树结构表示。本文提出的索引方法是为树中的每一个叶子结点建立两个DHT索引项。如果索引项的码是对应该叶子结点的XPath查询，那么值将是存储数据的网络节点的IP地址；如果索引项的码是对应该叶子结点的父结点的XPath查询，值将是对应该叶子结点的XPath查询。

查询的时候，查询请求将被转化为多个XPath查询表达式。对于一个XPath查询表达式，先在DHT中对整个查询表达式做精确匹配，如果没有结果返回，就去掉这个查询表达式的最后一个定位步骤生成其父查询表达式，然后在DHT中对父查询表达式进行精确匹配，最后再在父查询表达式的返回结果中进行查找。多个XPath查询表达式可以通过逻辑运算构成复杂的查询，对于这种复杂的查询的执行，系统需要进行优化，以便采用最好的执行方案。

在今后进一步的研究工作中，首先要考虑的是对本文提出的方法进行实验分析，进一步验证它的性能。其次，本文提出的方法还有待进一步完善：如如何有效地进行数据的修改和删除；又如还需要进一步讨论进行复杂查询时的进行查询优化的方法，评价它们的执行效率。

参考文献

1 Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of ACM SIGCOMM, 2001
2 Stocia I, Morris R, Karger, Kaashoek M, Balakrishnan H. Chord: A scalable peer-to-peer lookup services for internet applications. In: Proc. of the ACM SIGCOMM, 2001
3 Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proc. of Middleware, 2001
4 Zhao Y B, Kubiatowicz, Joseph A. Tapestry: An infrastructure for fault-tolerant wide-area location and routing: [Technical Report UCB/CSD-01-1141]. University of California at Berkeley, 2001
5 Harren M, Hellerstein J, Huebsch R, Loo B, Shenker S, Stoica I. Complex queries in DHT-based peer-to-peer networks. In: Proc. of IPTPS, 2002
6 Gupta A, Agrawal D, Abbadi A. Approximate range selection queries in peer-to-peer systems: [Technical Report UCSB/CSD-2002-23]. University of California at Santa Barbara, 2002

（上接第56页）

统对单个数据包需要完成从抓包、流量统计、流量分类、协议状态跟踪直到内容解码和关键字检索等一系列串行工作，耗时计算公式为 $Tpp = Tcp + Tnm + Tom + Tap$。其中 $Tcp \leqslant 1.25\mu s$，$Tnm \leqslant 0.5\mu s$，$Tom \leqslant 1\mu s$，在要求 $T \leqslant 5.3\mu s$ 的前提下，还剩余大约2.27μs可以供 Tap 消耗。而 Tap 期间内的主要工作是程序模块之间的调度和数据交互。

Tap 的一个重要时间开销在于操作系统由核心态(Kernel Mode)转为用户态(User Mode)所需要耗费的时间。由于在本实验环境下，Libpcap工作于操作系统的核心态，而其他程序模块运行在用户态，因此操作系统必须频繁地在核心态和用户态之间切换，每来回切换一次大约需要1200个CPU时钟周期，本例中换算成时间就是 $(1/2.4G) * 600 = 0.5\mu s$。因此，还剩余1.77μs可以用于监测程序的其他消耗。

系统的整体时间开销按照各个功能阶段划分如表5所示。

表5 网络监测系统的时间耗费分配表

功能	消耗时间	耗时比例
Libpcap	1.48μs	28%
基本网络监测	0.5μs	10%
应用监测	1μs	20%
调度开销和其余杂项开销	2.27μs	42%

为了验证系统的整体性能，我们将以上功能纳入统一项目，编译完成一个基本的网络监测系统，并在800Mbps的峰值流量下进行了整体连续运行和性能观察，结果程序运行正常，掉包率低于0.2%，CPU利用率稳定在75%～80%。

结论 经过分析和实验，证明了普通PC服务器完成千兆网络监测是完全可行的。当然本文提出的网络监测功能不多，也没有考虑监控和日志功能。随着网络带宽的不断增加，对实时监测的性能要求还会不断提高。本文提出的检测技术构架也存在很多可以改进的地方，例如：Libpcap本身的timestamp功能需要花费大量时间，可以进行精简；如果把监测系统程序全部放入操作系统核心态运行，也会大大改善系统性能。这些都是有待进一步研究和开发的课题。

参考文献

1 Baboescu F, Varghese G. Scalable packet classification. In: Proc. of ACM Sigcomm '01, Sep. 2001
2 Degioanni L, et al. Profiling and Optimization of Software-Based Network-Analysis Application
3 Agilent Technologies. JTC 003 Mixed Packet Size Throughput. http://advanced. comms. agilent. com/n2x/docs/journal/JTC_003. html
4 Buddhikot M M, Suri S, Waldvogel M. Space decomposition techniques for fast layer-4 switching. In: Proc. of PHSN, Aug. 1999
5 Eatherton W. Hardware-based internet protocol prefix lookups: [MS Thesis]. Washington University in St. Louis, Electrical Engineering Department, May 1999
6 Feldman A, Muthukrishnan S. Tradeofes for packet Classification. In: Proc. of Infocomm, March 2000
7 Gupta P, McKeown N. Packet Classification on Multiple Fields. In: Proc. ACM Sigcomm'99, Sept. 1999
8 Gupta P, Lin S, McKeown N. Routing lookups in hardware at memory access speeds. In: Proc. of the Conf. on Computer Communications (IEEE INFOCOMM), San Francisco, California March/April 1998, 3: 1241～1248
9 Gupta P, McKeown N. Packet Classification using hierarchical intelligent cuttings. In: Proc. of Hot Interconnects VII, Aug. 1999
10 Horspool R N. Practical fast searching in strings. Software - Practice & Experience, 1980, 10(6): 501～506
11 Baezz-Yates R A, Regnier, M. Average running time of the Boyer-Moore-Horspool algorithm. Theoretical Computer Science, 1992, 92(1): 19～31
12 Srinivasan V, Varghese G. Fast IP Lookups using Controlled Prefix Expansion. In: Proc. ACM Sigmetrics, June 1998
13 Srinivasan V, Suri S, Varghese G. Packet Classification using tuple space search. In: Proc. of ACM Sigcomm '99, Sept. 1999
14 Packet Classification Repository. http://www. cs. ucsd. edu/~baboescu/repository/
15 WinPcap Documentation. Available at: http://winpcap. polito. it/docs