文章编号:1001 - 9081(2009)02 - 0500 - 03

一种基于折半层次搜索的包分类算法

潘 登[1], 张大方[1], 谢 鲲[2], 张 继[1]
(1. 湖南大学 软件学院,长沙 410082; 2. 湖南大学 计算机与通信学院,长沙 410082)
(magic198449@163.com)

摘 要:折半层次搜索(BSOL)算法是一种高效的包分类算法,容易拓展至多维包分类,并支持 range 类型的规则。但由于其核心结构是在特里树(Trie)的每一层创建 hash 表,因此当 hash 装载因子较大或 hash 冲突较大时,会影响其效率。分析折半层次搜索算法的优缺点,引入布鲁姆过滤器,提出了一种新的改进算法,为 Trie 树的每一层建立了一个布鲁姆过滤器,在进行 hash 查找之前先进行一次布鲁姆查询运算,能够在 hash 冲突较大的情况下依然具有良好的性能。仿真实验结果表明,在数据包的命中率低于 90% 并且 hash 装载因子较大的情况下,新算法在运行时间上要优于以前的算法。

关键词:包分类;特里树;折半层次搜索;布鲁姆过滤器
中图分类号: TP393 **文献标志码**:A

An algorithm of packet classification based on binary search on levels

PAN Deng[1], ZHANG Da-fang[1], XIE Kun[2], ZHANG Ji[1]
(1. *College of Software Engineering, Hunan University, Changsha Hunan* 410082, *China*;
2. *College of Computer and Communication, Hunan University, Changsha Hunan* 410082, *China*)

Abstract: Binary Search On Levels (BSOL) is a fast algorithm for packet classification, which can be easily extended to multi-mention packet classification. For its core framework is hash Table which belongs to every layer of Trie Tree, the performance of BSOL will get lower when load factor of hash Table is big or hash collision is frequent. This paper presented a new algorithm using bloom filters. The new algorithm still performed well even the hash collision was frequent. Analyzing the experimental data from a virtual environment, the authors conclude that the new algorithm performs better when hit ratio of packets is lower than 90% while load factor is big.

Key words: packet classification; trie tree; binary search on levels; bloom filter

0 引言

随着计算机网络的飞速发展,越来越多的业务要求路由器能对数据包进行快速、准确地分类。研究有效的包分类算法及其实现技术是目前网络技术领域的热门课题[1]。

包分类是根据到达路由器的网络数据包的包头信息,通过匹配既定的包分类规则,对数据包进行分类处理。包分类的规则可能涉及数据包头的一个或多个维,如 IP 地址和端口。包分类规则有两种表示方式,一种是前缀表示方式,如 202.192.19.0/24,用来表示 202.192.19.0 到 202.192.19.255 的 IP 段;另一种是范围表示方式,如 202.192.19.5—202.192.19.7。包分类规则的两种表示方式各有优缺点:前缀表示方式容易被硬件接受,并且具有一些特殊的性质,如两个前缀只能有包涵或者不相交关系,但并非所有的 IP 段都能用一个前缀表示;范围表示方式虽然能表示所有的 IP 段,但难以被硬件实现[3]。

包分类算法从总体上可以分为两类:一类是靠纯硬件实现的算法,如内容存取记忆体(Content Access Memory, CAM),这类算法搜索速度很快,但由于 CAM 自身缺点(体积大,功耗大,不支持 range 类型规则)而难以被推广[2];另一类是通过软件实现的算法,细分为基于特里(Trie)树的算法和基于集合定位的算法。目前尚无一种算法能够在各方面都满足线速的要求。

本文提出的带布鲁姆过滤器的折半层次搜索(Binary Search on Levels based on Bloom Filters,BSOLBF)包分类算法,是在 BSOL(Binary Search on Levels)算法[3]思想的基础上提出的一种新的算法。该算法保持了 BSOL 算法的原有特性,并能在 hash 冲突较大或者 hash 装载因子过大的情况下,提升算法的性能。经过仿真测试,设定 hash 装载因子为0.75、0.69和0.8,采用规模为 30 万的规则库,BSOLBF 算法在搜索时间上要优于 BSOL 算法。当数据包在规则库中的命中率为 80% 时,运行时间约能提高 30%。

1 带布鲁姆过滤器的 BSOL 算法

BSOL 算法的核心结构在于为 Trie 树的每一层建立的 hash 表,但在 hash 冲突或者 hash 装载因子较大的情况下,通过 hash 运算判定一个元素不在 hash 表内是非常耗时的,极限情况下有可能要遍历整个 hash 表。关于 BSOL 算法的详细描述可以阅读参考文献[3]。本文提出的基于布鲁姆过滤器的 BSOL(BSOLBF)为 Trie 树的每一层建立了一个布鲁姆过滤

收稿日期:2008 - 08 - 05;**修回日期**:2008 - 09 - 23。 **基金项目**:国家自然科学基金资助项目(60673155,60703097);国家自然科学基金重大研究计划(90718008);国家 973 计划(2007CB310702);湖南省科技计划资助项目(2006GK3101)。

作者简介:潘登(1984 -),男,湖南长沙人,硕士研究生,主要研究方向:网络测试; 张大方(1959 -),男,上海人,教授,博士生导师,博士,CCF 高级会员,主要研究方向:可信系统与网络、容错计算; 谢鲲(1979 -),女,湖南长沙人,博士,主要研究方向:可信系统与网络、网络测试; 张继(1984 -),男,湖南长沙人,博士研究生,主要研究方向:网络测试。

器,在进行 hash 查找之前先进行一次布鲁姆运算,得到肯定结果后再进行 hash 查找,否则直接返回空结果。因此 BSOLBF 算法将在较大程度上缓解由于 hash 冲突而造成的算法性能下降。

1.1　布鲁姆过滤器

布鲁姆过滤器是一种空间效率很高而且具有很高可靠性的存储结构,用来判断一个元素是否在集合内。

为了表达 $S=\{e_1,e_2,\cdots,e_n\}$ 这样一个 n 个元素的集合,布鲁姆过滤器使用 k 个相互独立的 hash 函数,它们分别将集合中的每个元素映射到 $\{1,\cdots,m\}$ 的范围中。对任意一个元素 $e\in S$,第 i 个 hash 函数映射的位置 $h_i(e)$ 就会被置为 1($1\leqslant i\leqslant k$)。查询一个元素 x 是否在集合内时,首先通过 k 个 hash 函数得到 k 个映射位置 $h_i(x)$ ($1\leqslant i\leqslant k$),如果这 k 个映射位置全为 1 则说明元素 x 在集合内,否则就认为 x 不在集合内。

布鲁姆过滤器由于每一位都只能被赋值为 0 或 1,所以不支持对元素的删除操作。为了解决这个问题,人们采用了计数布鲁姆过滤器。其核心思想是为布鲁姆过滤器的所有位都各设置一个计数器,每当有元素通过 hash 运算被映射到某一位时,对应的计算器就加 1。当元素被删除时,就将该元素通过 k 个 hash 函数映射到的 k 个位的计数器全减 1,这样就不会因为删除操作而引起布鲁姆过滤器的假阴性错误。在 BSOLBF 算法中,考虑到需要动态添加删除规则,所以采用计数布鲁姆过滤器。

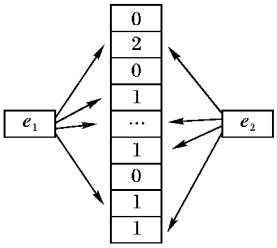

图 1　计数布鲁姆过滤器结构

与 hash 存储方法相比,布鲁姆过滤器最大的优势是它的空间效率,同时由于 Bloom Filter 不用处理 hash 冲突,因此它在增加或查找集合元素时所用的时间完全恒定,无论集合元素本身有多大。但布鲁姆过滤器的高效是建立在允许其发生低概率错误的基础上。当查询一个元素时,布鲁姆过滤器有很低的概率将不属于集合的元素误认为在集合中,这被称为假阳性错误。

1.2　BSOLBF 算法的实现

BSOLBF 算法实现分为两个部分:数据结构的建立和初始化;对网络数据包的匹配流程。分别如图 2、3 所示。

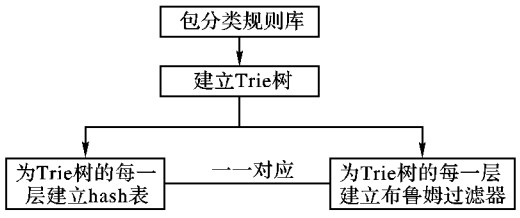

图 2　BSOLBF 算法所需的数据结构

BSOLBF 算法在实现上与 BSOL 相仿。首先根据包分类维数建立一棵 Trie 树,并将规则都挂接到 Trie 树的叶子节点上;然后为 Trie 树的每一层分别建立 hash 表和布鲁姆过滤器。利用 BSOLBF 算法进行查询时,先通过布鲁姆过滤器来确定目标节点是否在 hash 表内,得到肯定答案后再进行 hash 查找,否则直接向上折半搜索。所以当目标节点不在 hash 表内时,BSOLBF 算法仅用一次运算就能判定,而不需要遍历所有可能的 hash 冲突位置。这样就能在较大程度上缓解由于 hash 冲突而造成的算法性能下降。具体步骤如下列伪代码所示。

```
Algorithm buildBSOLBF( Rule) {
  root = new TrieNode;
  root. POList = Rule;
  root. bmr = NULL;
  if ( |root. POList| > T) split( root)
    Build hash Tables H0 H1 ...Hh for each layer
    Build Bloom Filters BF0 ...BFh for each layer
}
Algorithm split( TrieNode z) {
  z. left = new TrieNode;
  z. right = new TrieNode;
  z. left. bmr = z. right. bmr = z. bmr;
  for ( each rule r in z. POList) {
  if ( r contains z. left. Int and pri( r) > pri( z. left. bmr) )
    z. left. bmr = r;
  if ( r contains z. right. Int and pri( r) > pri( z. right. bmr) )
    z. right. bmr = r;
  }
  for ( each rule r in z. POList) {
    if ( r contains z. left. Int and pri( r) > pri( z. left. bmr) )
      z. left. bmr = r;
    if ( r contains z. right. Int and pri( r) > pri( z. right. bmr) )
      z. right. bmr = r;
  }
Algorithm BSOLBFSearch( d) {
  Left = 0;
  Right = 0;
  While( left < = right) {
  I = [( left + right) /2];
  Search BFi Using the First i bit of d;
  If ( Search BFi returns false)
  { right = i - 1;
    continue; }
  Search Hi Using the First i bit of d;
  If ( Search Hi returns null)
    right = i - 1;
  else if ( Search Hi returns a non-leaf node)
    left = i + 1;
  else if ( Search Hi returns a leaf node)
  { Search node. POList and
    return a best marching rule}
}
```

1.3　BSOLBF 算法分析

在查找时间方面,BSOLBF 算法利用了布鲁姆过滤器的特性与优点能在 hash 装载因子很大或者 hash 冲突很频繁的情况下拥有比 BSOL 更好的性能。在储存空间方面,BSOLBF 算法与 BSOL 算法相比,由于前者引入了布鲁姆过滤器,在储存开销上会有所增加。BSOL 算法的存储开销主要集中在 Trie 树的存储和每层 hash 表的存储上,其中 Trie 树的存储开销所占比重最大。BSOLBF 算法除了上述开销外还要额外的空间储存布鲁姆过滤器,但由于布鲁姆过滤器在空间上具有高效性(往往只需要 hash 存储空间的 1/8 ~ 1/4),所以 BSOLBF 算法在存储开销上的增加与 BSOL 算法相比只是很小幅度地增长。

BSOLBF 算法运用了布鲁姆过滤器作为规则的第一轮查

询,因此需要对布鲁姆过滤器的假阳性错误可能对算法性能造成的影响进行分析。

假定布鲁姆过滤器由 m 个比特的向量来表示,则任意一个比特被一个 hash 函数置 1 的可能性为 $1/m$,因此它没有被集合中所有 n 个元素置1的可能性为$(1-(1/m))^n$。由于每个元素都要通过 k 个 hash 函数将 k 个比特置1,所以该比特没有被置 1 的概率变成了$(1-(1/m))^{nk}$,那么它被置 1 的可能性即为$1-(1-(1/m))^{nk}$。对于布鲁姆过滤器的任意输入元素,被判定为在集合中的概率为 f:

$$f=\left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^k$$

当 m 很大时 f 可以进一步简化为:

$$f\approx(1-e^{\frac{-nk}{m}})^k$$

f 是与输入无关的,所表达的含义即为假阳性错误的发生概率。在 BSOLBF 算法中,假阳性错误如果发生,导致的结果就是实施一次返回结果为空的 hash 查找。用 E 来表示这种无效 hash 查找的次数,由于 BSOLBF 算法至多进行 $\log h$ 次布鲁姆过滤器运算,所以可以得到最坏情况下 E 的值。

$$E_{worst}=\log h+1$$

假设所有布鲁姆过滤器的假阳性错误率都是 f,那么在 $\log h$ 次布鲁姆过滤器运算中,可能会产生 $f\times\log h$ 次无效的 hash 查找。所以在平均情况下 E 值的上限可以表示为:

$$E_{avg}\leqslant f\log h+1$$

现假定通过设置 m 和 k 将每个布鲁姆过滤器的假阳性错误概率控制为0.001,在 $h=32$ 的情况下,进行10万次数据包的分类,至多发生 $0.001\times\log 32\times 100000=500$ 次由假阳性错误引起的无效 hash 查找。失效率为500/100000 = 0.005,采用布鲁姆过滤器作优化查询算法是有效的。

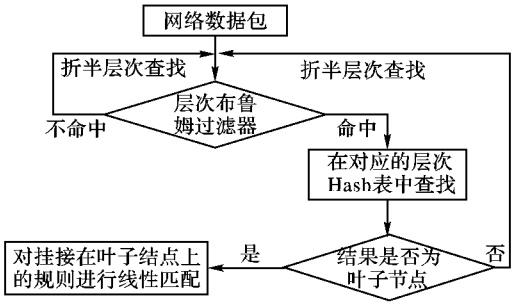

图3 BSOLBF 算法中数据包匹配流程

2 实验验证与结果分析

通过取自 CAIDA 网站的规则集(约 30 万条规则),对 BSOL 算法和 BSOLBF 算法进行了仿真对比实验。考虑到通过路由器的数据包不一定都能找到对应的包分类规则(实际应用中经常发生此种情况,路由器将会采取默认的方式处理该数据包),并且发生这种情况的概率对研究算法的性能有一定的影响,所以对比实验在数据包的命中率上给出了较小粒度的分析。

对比实验模拟了在不同的命中率下,BSOL 算法和 BSOLBF 算法分别处理 500 000 个到达路由器的数据包。图4~6分别是在 hash 装载因子 $a=0.8$,$a=0.75$ 和 $a=0.69$ 下的实验结果,以自制模拟平台下的运行时间为对比参数。运行环境为:Pentium4 3.06 G CPU,512 MB 内存,Windows XP 操作系统。

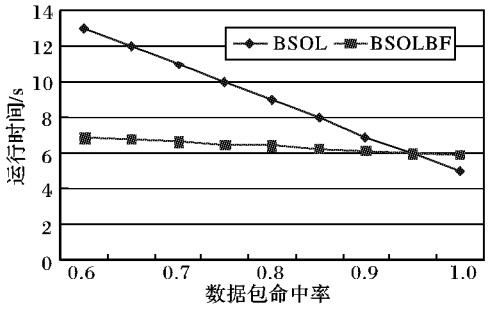

图4 $a=0.8$ 时的实验结果

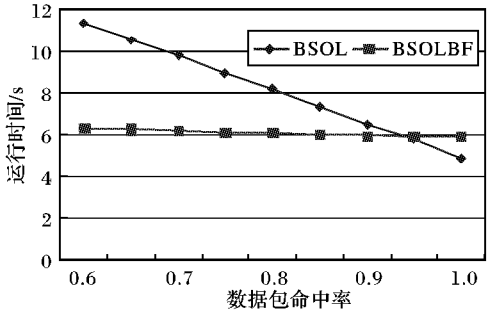

图5 $a=0.75$ 时的实验结果

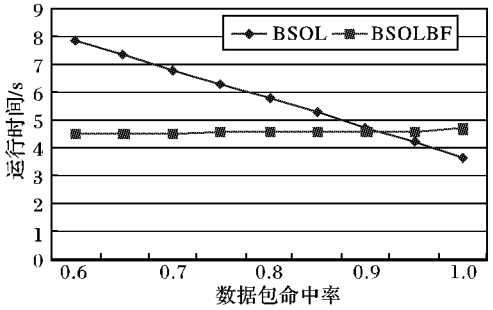

图6 $a=0.69$ 时的实验结果

由实验结果可以看出,随着数据包命中率的降低,BSOL 算法运行速度越来越慢,而 BSOLBF 算法的运行时间基本保持不变,这说明数据包的命中率对 BSOLBF 影响不大,而对 BSOL 算法的影响很大。造成这种现象的原因在于当数据包的命中率较小时,很多数据包在 hash 表内得不到正确匹配,需要遍历所有可能的 hash 冲突位置才能判定元素不在 hash 表内。在 hash 冲突或者 hash 装载因子较大的情况下,这种通过 hash 运算判定一个元素不在 hash 表内是非常耗时的,所以 BSOL 算法受数据包命中率的影响很大。而 BSOLBF 算法由于引入了布鲁姆过滤器,只需要一次操作就能判定元素是否在 hash 表内,所以数据包的命中率对它的影响很轻微。

在数据包的命中率为 100% 的情况下,BSOLBF 的运行时间要略长于 BSOL 算法,这是因为每个数据包都能在 hash 表内得到匹配,没有否定结果的存在,布鲁姆过滤器运算也就成了多余的操作。而事实上这种情形在实际应用不可能出现,尤其对于核心路由器来说,即便有再多的规则都不能保证通过路由器的所有数据包能全部得到匹配。

通过上述模拟实验可以得出的结论:在数据包的命中率低于 90% 并且 hash 装载因子偏大的情况下,BSOLBF 算法在运行时间上要优于 BSOL 算法。当数据包在规则库中的命中率为 80% 时,运行时间约能提高 30%。

(下转第 506 页)

功交易量逐步上升,在实验次数达到40次后逐渐保持平稳。表明DrTrust能准确反映节点信任值,有效提高交易的成功率。

4.2 DrTrust和EigenTrust的比较

初始条件与4.1节设置相同。两种信任模型分别模拟从善意点和恶意节点上的进行文件下载。仿真结果如图5所示。

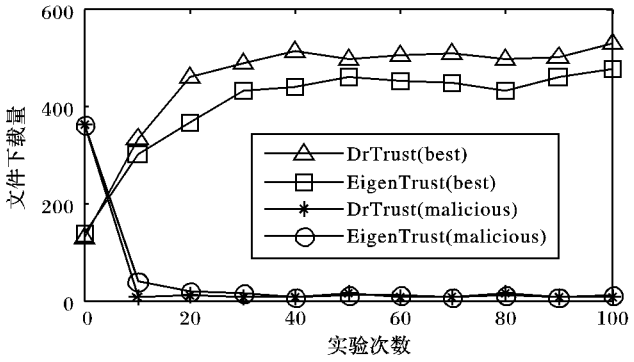

图5 DrTrust与EigenTrust性能比较

从图5可以看出,随着交易的进行,对于善意节点上文件的下载数量,DrTrust相对于EigenTrust有明显上升。而对于恶意节点上的文件下载数量,DrTrust比EigenTrust下降速度更快,并最终趋向于0。说明该模型相对于EigenTrust能更有效识别恶意节点,抑制恶意节点的行为,并能激励善意节点,提高成功交易率。

5 结语

本文从信任的定义、量化、存储和更新等方面出发,提出了一个应用于非结构化P2P网络中的信任模型DrTrust,强调了模型的可行性和合理性。仿真结果表明,该模型较好地实现了准确计算信任值和抑制恶意节点行为的功能。由于我们只采用了一个信任值衡量节点信任值,后续研究可以考虑将服务质量和节点信任结合起来衡量一个节点,并可以将DrTrust的思想用在结构化的P2P网络中。

参考文献:

[1] KALLATH, DINESH. Trust in trusted computing-the end of security as we know it[J]. Computer Fraud and Security, 2005(12): 4-7.

[2] KAMVAR S D, SCHLOSSER M T, GARCIA-MOLINA H. The Eigentrust algorithm for reputation management in P2P networks [C]// Proceedings of the 12th international World Wide Web conference. New York: ACM Press, 2003: 640-651.

[3] LI XIONG, LING LIU. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(7): 843-857.

[4] ZHOU RUNFANG, KAI H. PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2007, 18(4): 460-473.

[5] ABERER K, DESPOTOVIC Z. Managing trust in a peer-2-peer information system[C]// Proceedings of the 10th International Conference on Information and Knowledge Management. New York: ACM Press, 2001: 310-317.

[6] WALSH K, SIRER E G. Experience with an object reputation system for peer-to-peer file-sharing[C]// Proceedings of Symposium on Networked System Design and Implementation. Berkeley: USENIX Association, 2006: 1.

[7] ALMENAREZ F, MARIN A, DIAZ D, *et al*. Developing a model for trust management in pervasive devices[C]// Proceedings of Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops. Washington, DC: IEEE Computer Society, 2006: 267-271.

[8] GE LIANG, LUO JUNZHOU, XU YAOBIN. Developing and managing trust in peer-to-peer systems[C]// Proceedings of the 17th International Conference on Database and Expert Systems Applications. Washington, DC: IEEE Computer Society, 2006: 687-691.

[9] 孙知信, 唐益慰. 基于全局信任度的多层分组p2p信任模型[J]. 通信学报, 2007, 28(9): 133-140.

[10] SUN TAO, DENKO M K. A distributed trust management scheme in the pervasive computing environment[C]// Canadian Conference on Electrical and Computer Engineering. New York: IEEE, 2007: 1219-1222.

[11] JOSANG A, ISMAIL R, BOYD C. A survey of trust and reputation systems for online service provision[J]. Decision Support Systems. 2007, 43(2): 618-644.

[12] LIU YU-MEI, YANG SHOU-BAO, GUO LEI-TAO, *et al*. A distributed trust-based reputation model in p2p system[C]// Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. Washington, DC: IEEE Computer Society, 2007: 294-299.

[13] 陈贵海, 李振华. 对等网络: 结构、应用与设计[M]. 北京: 清华大学出版社, 2007.

(上接第502页)

3 结语

本文通过深入理解BSOL算法的核心思想,分析了该算法的优缺点,并提出了一种带布鲁姆过滤器的折半层次搜索包分类算法,即BSOLBF算法。BSOLBF算法继承了BSOL算法的原有特性,并能在hash冲突较大或者hash装载因子过大的情况下,提升算法的性能。经过仿真测试,设定hash装载因子为0.75和0.69,在规则集为30万条时,BSOLBF算法在搜索时间上要优于BSOL算法,数据包在规则库中的命中率越低优势越明显。

参考文献:

[1] 王永纲, 石江涛, 戴雪龙, 等. 网络包分类算法仿真测试与比较研究[J]. 中国科学技术大学学报, 2004, 34(4): 400-409.

[2] BABOESCU F, SINGH S, VARGHESE G. Packet classification for core routers: is there an alternative to CAMs[EB/OL]. [2008-06-23]. http://www.ieee-infocom.org/2003/papers/02_02.PDF.

[3] LU HAIBIN, SAHNI S. O(logW) multidimensional packet classification[C]// IEEE/ACM Transactions on Networking. New York: ACM, 2007: 462-472.

[4] DHARMAPURIKAR S, KRISHNAMURTHY P. Longest prefix matching using bloom filters[C]// IEEE/ACM Transactions on Networking. New York: ACM, 2006, (2): 201-212.

[5] ERGUN F, MITTRA V. A dynamic lookup scheme for bursty access patterns[C]// Proceedings of the IEEE INFOCOM. San Francisco: IEEE Computer Socity Press, 2001: 1444.

[6] 谢鲲, 张大方, 文吉刚, 等. 布鲁姆过滤器代数运算探讨[J]. 电子学报, 2008(5): 869.