面向NDN中名字查找的哈希布鲁姆过滤器

李 玮，张大方，徐 冰

(湖南大学信息科学与工程学院 长沙 410082)

【摘要】该文设计了一种面向NDN中名字查找的哈希布鲁姆过滤器(HBF)。HBF由位于片内存储器中的g个计数器布鲁姆过滤器(CBF)、g个计数器和位于片外存储器中的g个哈希表组成，每个哈希表与1个CBF和1个计数器关联。为了避免因部分CBF存入名字过多而导致HBF的高误判率，HBF通过二次哈希选择算法将NDN路由器中FIB/CS/PIT表项完整信息均匀分散保存于g个CBF和g个哈希表中，同时也利于数据包转发的并行处理。理论分析和实验结果表明在名字查找过程中，HBF利用片内存储器中CBF的定位与过滤作用，大幅度减少片外存储器的访问开销，提高数据包转发速率，有效避免泛洪攻击。

关 键 词 数据包转发速率； 哈希布鲁姆过滤器； 命名数据网络； 名字查找； 二次哈希选择算法

中图分类号 TP393 **文献标志码** A doi:10.3969/j.issn.1001-0548.2017.05.016

Hash Bloom Filters for Name Lookup in Named Data Networking

LI Wei, ZHANG Da-fang, and XU Bing

(College of Computer Science and Electronics Engineering, Hunan University Changsha 410082)

Abstract To provide quick name lookup technique, the paper designs a Hash bloom filter (HBF). The HBF consists of g on-chip counter bloom filters (CBFs), g on-chip counters and g off-chip Hash tables. Each Hash table is associated with a CBF and a counter. To reduce the false positive rate introduced by unbalanced name insertion in to CBFs, we propose two-Hash-choice algorithm which evenly disperses the FIB/CS/PIT entries into g Hash tables and CBFs. Moreover, HBF has a good feature of parallel processing of data packet forwarding because HBF adopts multiple Hash tables and CBFs. Theoretical and simulated results demonstrate that HBF can achieve very efficient name lookup by well utilizing the on-chip memory through localization and filtering function of CBF. Therefore, the proposed HBF improves data packet forwarding rate and effectively avoids flooding attacks.

Key words data packet forwarding rate; Hash bloom filter; named data networking; name lookup; two-Hash-choice algorithm

为了解决TCP/IP体系结构在路由扩展性、动态性、安全性、QoS、可靠性等方面日益突出的问题[1]，人们进行了大量研究，并取得了丰硕的研究成果，命名数据网络(named data networking, NDN)[2-3]就是其中的代表之一。NDN转发层中需要维护FIB (forwarding information base)、CS(content store)、PIT (pending interest table)3类信息。

可扩展的转发层是NDN广泛发展的关键，而FIB/CS/PIT中快速名字查找又是转发层的核心问题，特别是FIB与PIT不仅需要遵循最长前缀匹配(longest prefix matching, LPM)的规则进行名字查找，而且需要在大规模的名字集合中实现快速查找和更新，以满足路由器的传输速率。尽管传统网络体系中面向IP地址的最长前缀匹配算法已经非常成熟，但NDN命名特点使得名字查找比IP地址查找更加复杂；同时没有上限的名字空间造成路由器中路由表项数过多，空间急剧膨胀，这给NDN中名字存储和快速查找带来了巨大的挑战。目前针对NDN的名字查找技术有4种思路，分别是TCAM、哈希表、多步长字符特里树(multi-bit character trie)、布鲁姆过滤器(bloom filter, BF)。

文献[2]最早提出使用TCAM实现快速名字查找，但是由于一个名字的长度可能达到几百个字节，导致一个名字被拆分成多段存于TCAM中，因此需要多次TCAM查找，降低了查询速度，远远达不到IP地址查找时的效率[4]。

收稿日期：2016－01－05；修回日期：2016－07－08
基金项目：国家973项目(2012CB315805)；国家自然科学基金(61173167, 61472130)
作者简介：李玮(1972－)，男，博士，主要从事可信系统与网络、大数据处理等方面的研究.

文献[3,5]将CS、FIB、PIT分别存放于3个不同哈希表中，文献[6-7]采用线性链式哈希表和*d*-left哈希表等哈希技术来解决哈希冲突问题，减少查询时的访问次数。尽管哈希表具有$O(1)$的线性查找速度，但由于多个数据包达到时对同一个哈希表进行查询或更新操作，严重降低数据包的并发处理性能。同时由于哈希表占用空间较大，无法将CS/FIB/PIT等信息保存于访问速度较快但空间受到限制的SRAM中，只能保存于DRAM中，DRAM与SRAM(片内)访问延迟比为55:0.45[8]，当网络中出现大量泛洪攻击时，攻击包直接访问时延较高的DRAM，耗尽路由器内存资源，导致网络拥塞。

基于编码技术和特里树，文献[9-10]提出了名字词元编码特里树(name component encoding trie, NCET)或编码名字前缀特里树(encode name prefix trie, ENPT)来进行名字查找。但NCET或ENPT采用词元−编码映射表会增加额外存储空间、访问成本和名字词元分解成本。

为了压缩名字占用空间，文献[11-13]提出采用结构简洁和查询快速的BF来表示FIB或PIT，分别是DiPIT、UBF、Namefilter。但由于BF假阳性而无法进行有效回路检查；同时由于BF只能记忆元素是否属于某个集合，无法记忆元素详细信息，例如无法保存PIT时间戳等信息，这样对PIT中的过期表项就无法进行有效处理；UBF、DiPIT、Namefilter也未提及FIB、FIT中除了名字字段之外其余字段的存储设计方式。

为了有效解决上述问题，本文设计了一种面向NDN名字查找的哈希布鲁姆过滤器(HBF)。HBF由位于片内存储器中的g个计数器布鲁姆过滤器(counting bloom filter, CBF)、g个计数器和位于片外存储器中的g个哈希表组成。理论分析和实验结果表明HBF利用片内存储器中CBF的定位与过滤作用，大幅度减少片外存储器的访问开销，从而降低HBF的总体访问成本，提高数据包转发速率，有效避免泛洪攻击。通过理论和实验分析了HBF总体访问成本的影响因素，找出了最优参数设置，为工业界推广应用提供了理论设计依据。

1　相关工作

文献[16]首次提出利用BF加速IP地址查找。文献[13]据此提出Namefilter，直接使用名字前缀来代替IP前缀，将第二部分中哈希表换成n个BF。由于NDN中名字前缀集合数目不定，Namefilter中BF个数就无法确定，这就要求NDN路由器动态调整BF的个数。由于FPGA、ASIC等专用硬件不能支持运行时动态创建BF，造成该方法无法适用基于FPGA、ASIC的硬件平台。

文献[12]提出了基于BF的数据结构DiPIT，用于PIT的存储和快速查询及更新。DiPIT为NDN路由器中每个端口创建一个BF，用于存储经该端口的数据请求包的名字，同时创建一个共享BF，用来降低每个BF假阳性带来的误判。DiPIT中采用BF只能表示名字字段，无法表示PIT中每个表项的时间戳、Nonce列表、Face列表等字段。对于一些超过时限的PIT表项，DiPIT采用周期性衰减BF中每个计数器值的策略，会删除一些处于正常时限内的PIT表项，导致无法转发部分数据回复包。

为了有效存储PIT表项，文献[7]提出了一种基于*d*-left哈希表的PIT数据结构(*d*-left HTPIT)，当d=2时，*d*-left HTPIT由于占用空间大，只能存储于空间较大但访问时延较大的片外存储器(如DRAM)中，而每次名字查找平均访问次数为$d\left(\sum_{i=1}^{E} i \Big/ E\right)$，有时还需要查询overflow table，更加降低数据包处理效率。为了提高哈希表的并行处理速度，则需要不断增加d，但这样又会增加名字查找过程中对哈希表的访问次数。特别是遭遇泛洪攻击时，由于频繁的片外存储单元访问而导致NDN路由器宕机。

NDN的实现原型CCNx提出名字前缀哈希表(name prefix hash table, NPHT)[23]建立FIB和PIT共同的索引。FIB和PIT表项详细信息分别存于2个不同的哈希表中。NPHT最大优势是通过前缀之间的关联关系来提高最长前缀匹配效率。但NPHT存储了FIB或PIT名字的所有字符，内存空间占用较大，而且由于FIB与PIT索引存于同一个哈希表，这势必成为多个数据包并行处理时的访问瓶颈。

2　哈希布鲁姆过滤器

2.1　HBF结构和原理

与哈希表、树型存储及查询算法、Trie存储及查询算法等相比，BF所需要空间与元素自身大小无关，仅与元素个数相关，极大降低了存储空间。BF只能判断名字是否存在NDN路由器FIB/CS/PIT表中，而不能返回该名字对应的其他字段信息，因此需要用哈希表来存储组织FIB、PIT或CS表的详细信

息。由于哈希表较大，无法保存于片内存储器中，只能保存于片外存储器中，如DRAM。基于此，本文提出的HBF由g个CBF、g个计数器和g个哈希表组成，g个CBF和g个计数器存储于片内存储器中，如SRAM；g个哈希表存储于片外存储器中，如DRAM。其中哈希表中每个Entry由Key和Data两部分组成，Key代表名字，Data代表该名字对应的其他字段信息。HBF结构如图1所示。

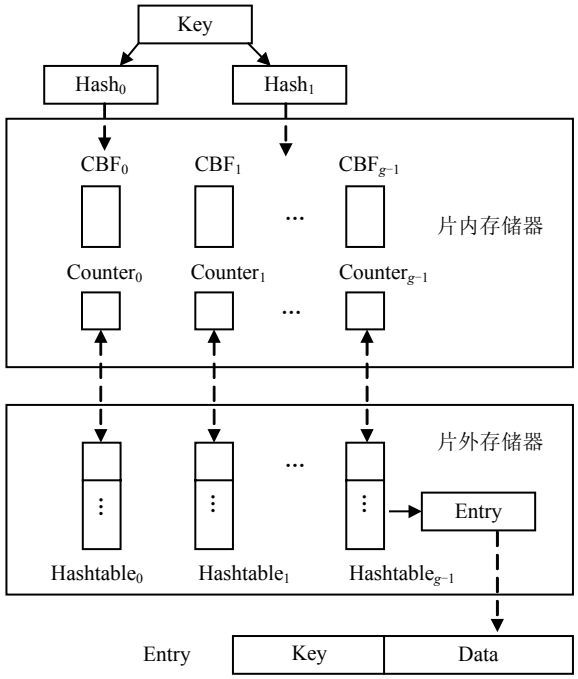

图1 HBF结构示意图

为了提高NDN路由转发并行处理效率，利用3个HBF分别为CS、FIB、PIT建立存储结构，而不是将CS、FIB、PIT信息存于同一个HBF中。当HBF应用于CS时，哈希表中每个Entry的Data代表数据内容(Content)；当HBF应用于FIB时，哈希表中每个Entry的Data代表FIB的转发规则，即Face列表；当HBF应用于PIT时，哈希表中每个Entry的Data代表请求Face列表、Nonce列表和期限时间戳等字段。

为了解决每个CBF和哈希表中插入名字个数不均衡的问题，HBF采用二次哈希的方法选择CBF和哈希表来保存名字及对应信息。下面以PIT中的名字插入和查询为例来说明HBF工作原理。

当有一个新的数据请求包达到时，CS中未能查询到请求数据内容，同时PIT中也未发现该数据请求记录，因此需要向PIT中插入该条数据请求记录，插入过程分为3步：

1) 利用两个哈希函数计算该数据请求包中名字字段的哈希值，分别为$Hash_0$和$Hash_1$；

2) 查询$Hash_0$和$Hash_1$对应的两个计数器$Counter_i$和$Counter_j$的值，如果$Counter_i$>$Counter_j$，则将该名字插入到CBF_j中，否则插入CBF_i中；

3) 如果$Counter_i$≤$Counter_j$，则将该名字及其他信息插入$Hashtable_j$中，否则插入$Hashtable_i$中。

当数据回复包达到NDN路由器时，需要从PIT中查询数据请求Face列表，查询过程分为3步：

1) 利用两个哈希函数计算该数据回复包中名字的哈希值，分别为$Hash_0$和$Hash_1$；

2) 分别查询$Hash_0$和$Hash_1$对应的CBF_i和CBF_j中是否存在该名字，可能出现4种判断结果：① CBF_i判断存在，CBF_j判断不存在；② CBF_i判断不存在，CBF_j判断存在；③ CBF_i和CBF_j判断都存在；④ CBF_i和CBF_j判断都不存在；

3) 根据上述4种判断结果，对哈希表的查询操作分别进行如下处理：

① CBF_i判断存在，CBF_j判断不存在。进入$Hashtable_i$中查询数据回复包中名字对应的其他信息，如果能查询到该名字，则读取数据请求Face列表进行数据转发；如果未能查询到该名字，说明CBF_i产生误判，不做任何处理；

② CBF_i判断不存在，CBF_j判断存在。进入$Hashtable_j$中查询数据回复包中名字对应的其他信息，如果能查询到该名字，则读取数据请求Face列表进行数据转发；如果未能查询到该名字，说明CBF_j产生误判，不做任何处理；

③ CBF_i和CBF_j判断都存在。进入$Hashtable_i$中查询数据回复包中名字对应的其他信息，如果能查询到该名字，则读取数据请求Face列表进行数据转发，流程结束；如果未能查询到该名字，说明CBF_i产生误判，进入$Hashtable_j$中查询数据回复包中名字对应的其他信息，如果能查询到该名字，则读取数据请求Face列表进行数据转发；如果未能查询到该名字，说明CBF_j产生误判，不做任何处理；

④ CBF_i和CBF_j判断都不存在。说明该数据回复包不是该NDN路由器请求的，直接丢弃该数据包，不做任何处理。特别针对泛洪攻击，由于CBF的过滤作用，避免直接进入位于片外存储器中的$Hashtable_i$或$Hashtable_j$查询，从而有效防止因泛洪攻击造成的NDN路由器内存耗尽和宕机。

2.2 HBF算法分析

本节主要对HBF算法空间复杂度和时间复杂度

的影响因素进行理论分析。时间复杂度主要是指名字查询过程时对片内存储器中CBF的访问次数和片外存储器中哈希表的访问次数，二者均受到CBF误判率的影响。首先需分析HBF中多个CBF组合在一起后的误判率。设定HBF中CBF与哈希表个数均为g，名字最大个数为d，每个CBF和哈希表保存名字的个数为$n=d/g$。每个CBF具有m个计数器和k个哈希函数，每个计数器具有t个比特。

1) 误判率分析(假阳性)

HBF中，每个名字的2个哈希值对应的2个CBF同时不出现假阳性时才不会产生误判现象。根据文献[15]，HBF误判率的计算公式为：

$$f_{\text{HBF}}=1-(1-f_{\text{CBF}})^2=1-(1-\mathrm{e}^{-kn/m})^2 \tag{1}$$

2) 空间复杂度分析

HBF将CBF和哈希表分别部署在片内和片外存储器中，设定HBF总占用空间为S_{HBF}，CBF占用空间为S_{CBF}，哈希表占用空间为S_{HT}，FIB/CS/PIT每条记录为S_{Entry}字节，内存占用空间为：

$$S_{\text{HBF}}=S_{\text{CBF}}+S_{\text{HT}}=d\frac{m}{n}\frac{t}{8}+dS_{\text{Entry}} \tag{2}$$

从式(2)看出给定m值，S_{CBF}只与名字个数有关，与名字自身长度无关，这极大压缩了名字占用空间，保证片内存储器可容纳更多名字个数。

3) 片内存储器访问次数分析

设定待查找名字个数为q，HBF中存储名字个数与q的比例为r，误判个数为$(q-qr)f_{\text{HBF}}$。则$qr+(q-qr)f_{\text{HBF}}$个名字每次需要访问k次CBF，$q(1-r)(1-f_{\text{HBF}})$个名字每次可能需要访问1~k次CBF。以k=4为例，在未产生误判情况下，对于每个名字在CBF中的4个哈希判断结果组合共有15个。其中判断结果为0000、0001、0010、0011、0100、0101、0110、0111只需要访问1次CBF，其概率为8/15；判断结果为1000、1001、1010、1011需要访问2次CBF，其概率为4/15；判断结果为1100、1101需要访问3次CBF，其概率为2/15；判断结果为1110需要访问4次CBF，其概率为1/15。判断不在单个CBF中名字查找平均访问次数$V_{\text{CBF-1}}$为：

$$V_{\text{CBF-1}}(k=4)=\frac{(q-qr)}{q}\left(\frac{8\times1+4\times2+2\times3+1\times4}{15}\right)=$$

$$(1-r)\frac{1}{2^4-1}\sum_{i=1}^{4}(2^{4-i}i) \tag{3}$$

以此类推，片内存储器的平均访问次数$V_{\text{CBF-2}}$计算公式为：

$$V_{\text{CBF-2}}=\frac{[qr+(q-qr)f_{\text{HBF}}]k+2(q-qr)(1-f_{\text{HBF}})\dfrac{\sum_{i=1}^{k}(2^{k-i}i)}{2^k-1}}{q}=$$

$$[r+(1-r)(1-(1-\mathrm{e}^{-kn/m})^2)]k+2(1-r)[1-(1-(1-\mathrm{e}^{-kn/m})^2)]\frac{\sum_{i=1}^{k}(2^{k-i}i)}{2^k-1} \tag{4}$$

从式(4)可看出，$V_{\text{CBF-2}}$与HBF片内存储器占用空间m呈单调下降关系，即m越大，$V_{\text{CBF-2}}$越低。在固定HBF片内存储器占用空间条件下，设定m/n=10，片内存储器的平均访问次数$V_{\text{CBF-2}}$与参数r、k相关。

4) 片外存储器访问次数分析

哈希表(链地址)的装填因子为a，对于HBF，同一个名字查找可能要遍历2个哈希表，需综合2个哈希表来计算HBF总体平均查找长度(次数)。存储于HBF中有50%名字在第1个哈希表中查询到结果后就退出查询，不再进入第2个哈希表进行查询；50%名字在第1个哈希表查找失败后再次在第2个哈希表中查询得到结果。使用CBF后，进入哈希表查找名字个数为真正存储于HBF中名字个数与CBF误判名字个数之和，其访问次数$V_{\text{CBF-HT}}$计算公式为：

$$V_{\text{CBF-HT}}=\frac{\left(qrS_{\text{nc}}+\dfrac{qr}{2}U_{\text{nc}}\right)+2(q-qr)\dfrac{f_{\text{HBF}}}{2}U_{\text{nc}}}{q}=$$

$$r\left(1+a+\frac{\mathrm{e}^{-a}}{2}\right)+(1-r)(1-(1-\mathrm{e}^{-kn/m})^2)(a+\mathrm{e}^{-a}) \tag{5}$$

未使用CBF过滤时，所有名字查找时都将直接访问哈希表，其访问次数V_{HT}计算公式为：

$$V_{\text{HT}}=\frac{\left(qrS_{\text{nc}}+\dfrac{qr}{2}U_{\text{nc}}\right)+2(q-qr)U_{\text{nc}}}{q}=$$

$$r\left(1+a+\frac{\mathrm{e}^{-a}}{2}\right)+2(1-r)(a+\mathrm{e}^{-a}) \tag{6}$$

5) 总体访问成本分析

通过上述理论分析可看出在给定CBF片内存储器占用空间、r、a情况下，HBF中片内存储器访问

次数随着k增加而增加。当$k<k_0$时，片外存储器访问次数则随着k增加而大幅度降低；当$k=k_0$时，片外存储器访问次数取得最小值；当$k>k_0$时，片外存储器访问次数则随着k增加而稍微增加。但片外存储器访问时间要远高于片内存储器，因此需要分析HBF的总体访问成本(综合考虑片内和片外存储器访问次数)。设定片内存储器每次访问成本为τ，片外存储器每次访问成本为$\beta\tau$，则HBF总体访问成本$\mathrm{Cost}_{\mathrm{HBF}}$计算公式为：

$$\mathrm{Cost}_{\mathrm{HBF}}=\tau V_{\mathrm{CBF\text{-}2}}+\beta\tau V_{\mathrm{CBF\text{-}HT}} \tag{7}$$

未采用CBF直接访问哈希表成本计算公式为：

$$\mathrm{Cost}_{\mathrm{HT}}=\beta\tau V_{\mathrm{HT}} \tag{8}$$

HBF采用SRAM和DRAM分别存储CBF和哈希表，给定m/n=10，a=1，取τ=1，β=55/0.45=122[18]，则根据式(7)和式(8)得到$\mathrm{Cost}_{\mathrm{HT}}$与$\mathrm{Cost}_{\mathrm{HBF}}$比较结果如表1所示。$r$=0.5时，$\mathrm{Cost}_{\mathrm{HBF}}$约为$\mathrm{Cost}_{\mathrm{HT}}$的30.8%；随着NDN路由器运行时间越久，$r$会逐步降低，而随着$r$降低，CBF总体访问成本降低更加明显，$r$=0.01时，$\mathrm{Cost}_{\mathrm{HBF}}$约为$\mathrm{Cost}_{\mathrm{HT}}$的2.5%。

表1 $\mathrm{Cost}_{\mathrm{HT}}$与$\mathrm{Cost}_{\mathrm{HBF}}$理论对比

r	0.01	0.1	0.2	0.3	0.4	0.5
$\mathrm{Max}(\mathrm{Cost}_{\mathrm{HT}}/\mathrm{Cost}_{\mathrm{HBF}})$	40.55	13.69	7.78	5.37	4.07	3.25

根据式(4)、式(5)、式(7)可以看出，在选定片内存储器和片外存储器后，HBF总体访问成本$\mathrm{Cost}_{\mathrm{HBF}}$与$m/n$、$r$、$k$、$a$等参数相关，相互关系如下：

① $\mathrm{Cost}_{\mathrm{HBF}}$与$m/n$是单调减的关系，即$\mathrm{Cost}_{\mathrm{HBF}}$随着$m/n$增加而减小，但会增加CBF的占用空间；

② $\mathrm{Cost}_{\mathrm{HBF}}$与$r$是单调增的关系，即$\mathrm{Cost}_{\mathrm{HBF}}$随着$r$减小而减小，路由器运行时间越长，$r$会越小；

③ $\mathrm{Cost}_{\mathrm{HBF}}$与$a$是单调增的关系，即$\mathrm{Cost}_{\mathrm{HBF}}$随着$a$减小而减小，但会增加哈希表的占用空间；

④ $\mathrm{Cost}_{\mathrm{HBF}}$与$k$既有单调增的关系，也有单调减的关系，$k<k_0$时，$\mathrm{Cost}_{\mathrm{HBF}}$与$k$是单调减的关系，$k>k_0$时，$\mathrm{Cost}_{\mathrm{HBF}}$与$k$是单调增的关系。

通过上述关系分析，在固定占用空间的情况下，k是决定$\mathrm{Cost}_{\mathrm{HBF}}$大小的关键参数，特别是$k_0$的选择，这给工业界的推广应用提供了理论依据。

3 实验评估

3.1 实验方案及数据

实验主要目标是验证理论分析正确性，找出最优参数设置，优化HBF总体访问成本，降低访问开销，提高NDN数据包转发速率。同时将HBF的访问访问成本与d-left HTPIT对比分析。

实验数据有两个途径。1) 从Blacklist[19]下载学术界广泛使用的域名和URL集合，从URL解析出域名后并重新生成名字集；2) 利用文献[20]开发的NDN数据生成工具NDNBench，以Blacklist下载的URL集合为种子，随机生成多组名字集合。

实验数据以Blacklist子目录Port中URL集为种子，利用NDNBench生成50组查询名字集，每个查询集包括1 000 000个名字，然后分别抽取查询集中0.1%、1%、10%的元素构成插入名字集(即r=0.001，r=0.01，r=0.1)。查询集或插入集中名字对应的其他字段信息随机生成。

3.2 实验结果分析

1) HBF实际总体访问成本对比分析

根据上述实验，计算HBF和直接访问哈希表实际总体访问成本(以1 000个名字为统计单位)，$\mathrm{Cost}_{\mathrm{HT}}$与$\mathrm{Cost}_{\mathrm{HBF}}$实际结果对比如表2所示。$r$=0.1时，$\mathrm{Cost}_{\mathrm{HBF}}$约为$\mathrm{Cost}_{\mathrm{HT}}$的8.2%；随着$r$降低，CBF总体访问成本降低更加明显，$r$=0.001时，$\mathrm{Cost}_{\mathrm{HBF}}$约为$\mathrm{Cost}_{\mathrm{HT}}$的2.1%。

表2 $\mathrm{Cost}_{\mathrm{HT}}$与$\mathrm{Cost}_{\mathrm{HBF}}$实际结果对比

r	0.001	0.01	0.1
$\mathrm{Max}(\mathrm{Cost}_{\mathrm{HT}}/\mathrm{Cost}_{\mathrm{HBF}})$	47.9	33.02	12.15

2) HBF与d-left HTPIT访问次数及成本对比

d-left HTPIT中参数d(哈希表个数)越大时，数据包的并发处理对哈希表的访问效率就越高，但名字查找时需要遍历d个哈希表，片外存储器的访问次数就会大幅度上升。

HBF片内存储器和片外存储器的访问次数与其参数g(CBF与哈希表的个数)无关，当g取值越大时，数据包的并发处理时对哈希表的访问效率就越高。HBF以牺牲片内存储器空间为代价，通过片内存储器中的CBF减少对片外存储器中哈希表的无效访问次数。根据文献[15]可知，当$kn\geq m$，CBF的误判率(假阳性)f_{CBF}接近1，全部元素会被误判，导致CBF失效，因此会有$m/n>k$。一般k最小值取2，因此当m/n=3时，HBF占用最小的片内存储存储器空间，此时代价最低，即每个名字消耗12 bits(1.5 byte)。

将具有最低片内存储空间的HBF与具有最低哈希表个数的d-left HTPIT进行对分析，如表3所示。

表3　HBF与*d*-left HTPIT访问次数及访问成本对比

算法	片外存储器访问次数	总体访问成本
HBF	625.47	79 173.79
d-left HTPIT	2 446.03	298 415.51

从表3可以看出，HBF在占用最小片内存储空间情况下，其片外存储器访问次数和总体访问成本约为*d*-left HTPIT的25%。

将HBF的片内存储空间提高到m/n=10(每个名字消耗40 bits)后，再与*d*-left HTPIT(*d*=2)对比分析，其实验结果如图2、图3所示。

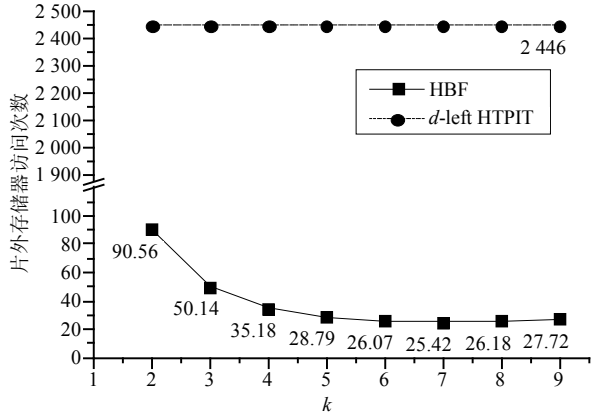

图2　片外存储器访问次数比较(r=0.001)

图2可看出k=2时，HBF片外存储器访问次数约为*d*-left HTPIT的3.3%；k=7时，HBF片外存储器访问次数约为*d*-left HTPIT的1%。

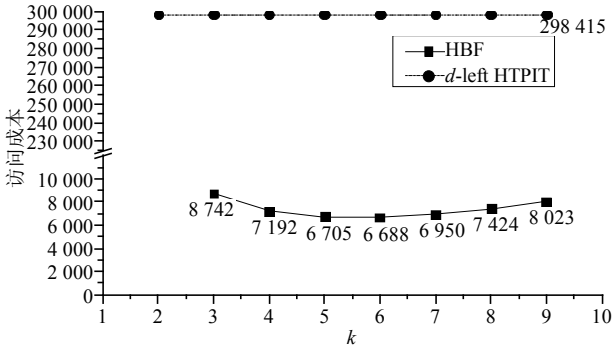

图3　总体访问成本比较(r=0.001)

图3可看出同*d*-left HTPIT相比，尽管HBF增加了片内存储器的访问次数，但总体访问成本还是显著降低，k=2时，约为*d*-left HTPIT的5%；k=5时，约为*d*-left HTPIT的2.5%。

4　结 束 语

本文提出了一种名为哈希布鲁姆过滤器的数据结构及相应查询算法，该结构通过CBF的定位与过滤作用，避免查找时对g个哈希表的遍历操作，大幅度减少对片外存储器的访问开销，降低名字查找的总体访问成本，提高名字查找速率，有效避免泛洪攻击。本文对HBF总体访问成本$\mathrm{Cost}_{\mathrm{HBF}}$与$m/n$、$r$、$k$、$a$等参数关系进行了系统理论分析和实验验证，为工业界应用提供了设计依据。

同时通过与类似研究成果*d*-left HTPIT对比，HBF在NDN名字查找过程的内存访问次数(片外存储器)和总体访问成本大幅度降低，在其占用最少片内存储器空间情况下(每个名字消耗12 bits)，片外存储器访问次数和总体访问成本约为*d*-left HTPIT的25%；当其占用空间提高到每个名字消耗40 bits时，片外存储器访问次数约为*d*-left HTPIT的1% (HBF中k=7)。而这样的比较结果还是在*d*-left HTPIT中哈希表个数d设为最小值时取得的，此时*d*-left HTPIT中哈希表会成为数据包并发处理时资源访问的瓶颈。为了解决此问题则需要提高哈希表个数d，那么HBF在总体访问成本的优势就会更加突出。

参 考 文 献

[1] 谢高岗, 张玉军, 刘韵洁, 等. 未来互联网体系结构研究综述[J]. 计算机学报, 2012, 35(6): 1109-1119.
XIE Gao-gang, ZHANG Yu-jun, LIU Yun-jie, et al. A survey on future internet architecture[J]. Chinese Journal of Computers, 2012, 35(6): 1109-1119.

[2] ZHANG L, ESTRIN D, JACOBSON V, et al. Named data networking (ndn) project. in Technical Report, NDN-0001, 2010[EB/OL]. [2010-10-31]. http://www.named-data.net/.

[3] JACOBSON V, SMETTERS D K, THORNTON J D, et al. Networking named content[C]//Proceedings of International Conference on Emerging Networking Experiments and Technologies. Rome, Italy: IEEE, 2009: 1-12.

[4] 汪漪. 内容中心网络路由查找关键技术研究[D]. 北京: 清华大学, 2013.
WANG Yi. Research on name lookup in named data networking[D]. Beijing: Tsinghua University, 2013.

[5] YUAN H, SONG T, CROWLEY P. Scalable NDN forwarding: Concepts, issues and principles[C]//Proceedings of International Conference on Computer Communication Networks. Munich, Germany: IEEE, 2012: 1-9.

[6] MATTEO V, DIEGO P, LEONARDO L .On the design and implementation of a wire-speed pending interest table[C]//Proceedings of IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking. Turin, Italy: IEEE, 2013: 1-6.

[7] YUAN Hao-wei, CROWLEY P. Scalable pending interest table design: from principles to practice[C]//Proceedings of IEEE International Conference on Computer Communications. Toronto, Canada: IEEE, 2014: 2049-2057.

[8] WEI You, BERTRAND M, PATRICK T, et al. Realistic storage of pending requests in content-centric network

routers[C]//Proceedings of the 1st IEEE International Conference on Communications in China: Communications QoS and Reliability. Beijing, China: IEEE, 2012: 121-125.

[9] WANG Yi, HE Ke-qiang, LIU Bin, et al. Scalable name lookup in NDN using effective name component encoding[C]//Proceedings of International Conference on Distributed Computing Systems. Macau, China: IEEE, 2012: 688-696.

[10] DAI H, LIU B, CHEN Y, et al. On pending interest table in named data networking[C]//Proceedings of ACM/IEEE Architectures for Networking and Communications Systems. Austin, Texas, USA: IEEE, 2012: 211-222.

[11] LI Z, BI J, WANG S. Compression of pending interest table with bloom filter in content centric network[C]// Proceedings of ACM International Conference on Future Internet Technologies. Seoul, Korea: ACM, 2012: 47.

[12] WEI You, BERTRAND M, PATRICK T, et al. DiPIT: a distributed bloom-filter based PIT table for CCN Nodes[C]//Proceedings of IEEE International Conference on Computer Communications and Networks. Munich, Germany: IEEE, 2012: 1-7

[13] WANG Yi, PAN Tian, LIU Bin, et al. NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom filters[C]//Proceedings of IEEE International Conference on Computer Communications, Mini-conference. Turin, Italy: IEEE, 2013: 95-99.

[14] BlOOM B. Space/time trade-offs in hash coding with a llowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426.

[15] BRODER A, MITZENMACHER M. Network applications of bloom filters: a survey[J]. Internt Mathematics, 2005, 1(4): 485-509.

[16] SARANG D, PRAVEEN K, DAVID E T. Longest prefix matching using bloom filters[C]//Proceedings of ACM International Conference on the Applications, Technologies, Architectures, and Protocols for Computer Communication. Karlsruhe, Germany: ACM, 2006: 201-212.

[17] 严蔚敏, 吴伟明. 数据结构[M]. 北京: 清华大学出版社, 2011.
YAN Wei-min, WU Wei-ming. Data structure[M]. Beijing: Tsinghua University Press, 2011.

[18] PERINO M D, VARVELLO. A reality check for content contric networking[C]//ACM SIGCOMM Workshop on Information-Centric Networking. Toronto, Canada: ACM, 2011: 44-49.

[19] URLBLACKLIST. Blacklist data set[EB/OL]. [2014-09-23]. http://www.urlblacklist. com/.

[20] ZHANG Ting, WANG Yi, LIU Bin, et al. NDNBench: a benchmark for named data networking lookup[C]// Proceedings of IEEE Global Communications Conference, incorporating the Global Internet Symposium. Atlanta, GA, USA: IEEE, 2013: 2152-2157.

编 辑 蒋 晓