ISSN 1000-0054　清华大学学报（自然科学版） 2018 年 第 58 卷 第 1 期　1/17
CN 11-2223/N　J Tsinghua Univ (Sci & Technol), 2018, Vol. 58, No. 1　1-7

NDN 名字查找算法的性能测试平台的设计和实现

张　庭[1]，汪　漪[2]，杨　仝[3]，卢建元[1]，刘　斌[1]

（1. 清华大学 计算机科学与技术系，北京 100084；2. 华为未来网络理论实验室，香港 999072；
3. 北京大学 信息科学技术学院，北京 100871）

摘　要：在内容标记网络(NDN)中，越来越多的名字查找算法被提出。这些算法的性能包括速度、可扩展性以及更新性能等亟需评估。但是，NDN 目前还处在研究阶段，没有大规模 NDN 网络部署，缺少真实的大规模名字查找表以及相应的流量。该文设计并实现了一个用于评测名字查找算法性能的软件测试平台——NDNBench。NDNBench 包含 4 个部分：种子表分析器、名字表产生器、名字流量产生器以及更新流量产生器。实验发现名字表和流量的特征会在很大程度上影响 NDN 名字查找的性能。NDNBench 平台将这些特征进行提取，形成可以量化的参数并提供给用户。用户通过调整 NDNBench 的不同参数，可以得到具有不同结构特征以及表项数目的名字查找表和相应测试流量，从而对名字查找算法性能进行测试评估。该文还对现有的一些名字查找算法进行评估。NDNBench 已经在最近的一些工作中得到应用。

关键词：内容标记网络(NDN)；名字查找；性能评测

中图分类号：TP393.0　　**文献标志码**：A

文章编号：1000-0054(2018)01-0001-07

DOI：10.16511/j.cnki.qhdxxb.2018.22.001

Design and implementation of an evaluation platform for NDN name lookup algorithms

ZHANG Ting[1], WANG Yi[2], YANG Tong[3], LU Jianyuan[1], LIU Bin[1]

(1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
2. Huawei Future Network Theory Laboratory, Hong Kong 999072, China;
3. School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: Many name lookup algorithms have been proposed for named data networking (NDN). These algorithms need to be evaluated based on their reachable speed, scalability, and update performance. However, NDN is still in the research stage so there are no large NDN networks and no large real name routing tables or NDN traffic. This paper presents a software test platform, NDNBench, to evaluate, compare and test different name lookup algorithms. NDNBench consists of a seed forwarding information base (FIB) analyzer, an FIB generator, a name trace generator and an update generator. Tests show that the name table and traffic characteristics greatly influence the NDN name lookup algorithm performance. The platform extracts these features, forms quantifiable parameters and provides them to the user. The parameters of NDNBench can be adjusted to obtain various FIBs and traces with structure and size diversity to test the lookup algorithms. This paper also evaluates some existing name lookup algorithms.

Key words: named data networking (NDN); name lookup; performance evaluation

互联网最初的目标是追求网络的互联以实现硬件资源的共享。随着技术的进步以及信息化的普及，互联网的功能正在逐渐演变。用户不再关心内容存储的位置，而关注内容本身的易获取性。内容标记网络(named data networking，NDN)作为未来网络架构被提出，并得到了越来越多的研究[1-5]。NDN 将数据本身和数据存储的物理位置分离开来，并且使用名字进行查找，这就避免了内容与其位置的映射，提高了网络数据检索的效率。

基于名字的路由查找是 NDN 网络中的一项关键技术。NDN 网络中传输的数据包分为两类：数据请求包和数据回复包。当数据请求包到达 NDN 路由器后，请求包中请求的名字将作为关键字首先在路由器的内容缓存表和请求状态表中进行查找。如果没有找到，则在名字查找表(下文简称名字表)

收稿日期：2017-02-27

基金项目：国家自然科学基金重点项目(61432009)；
国家自然科学基金面上项目(61373143)；
国家自然科学基金青年项目(61602271)；
中国博士后科学基金项目(2016M591182)；
高等学校博士学科点专项科研基金项目
(20130002110084)

作者简介：张庭(1990—)，男，博士研究生。

通信作者：刘斌，教授，E-mail：liub@tsinghua.edu.cn

中进行最长前缀匹配查找来得到下一跳端口号，继而将数据包转发到相应的端口。NDN 基于名字进行数据包的查找比 IP 网络路由查找更有挑战性，主要体现在以下几个方面：1）NDN 中内容名字比 IP 地址更为复杂。与采用定长 32 比特的 IP 地址不同，NDN 名字采用类似于 URL 的名字结构，长度可变并且没有上限。2）NDN 名字表将比现有的 IP 路由表至少大 1 至 2 个数量级。现有 IP 路由表大约有 6×10^5 个前缀表项，而名字表项数目可能达到百万级别甚至更高。3）NDN 名字表更新频率比现有路由表更高。这是因为除了网络拓扑变化以及路由策略变化导致表项更新以外，在 NDN 场景中，内容的添加和删除也会导致名字表的更新，而且预期这种操作导致的更新会越来越频繁。

已有的 IP 路由查找算法不能直接运用到 NDN 场景中，需要根据 NDN 名字表的结构提出新的查找算法。一些名字查找算法[2-5]已经被提出来解决这个问题。为了评测和比较这些算法以及未来提出的其他算法的性能，迫切需要一个独立的测试平台。但是，由于 NDN 目前还处在初步研究阶段，还没有真实的大规模的名字表，也无法在网络中截取 NDN 流量来对名字查找算法进行测试。

另外，关于查找算法的性能测评之前已经有研究[6-11]。Taylor 等[6]开发了针对五元组流分类算法的性能测评工具。FRuG 平台[7]分析并且产生 IPv4 前缀及 MAC 地址，类似的工作还有 Castelino 等提出的转发表(forwarding information base，FIB)生成工具[8]。Wang 等[9]以及 Zheng 等[10]针对 IPv6 路由表进行了性能测评。但是，这些研究都是针对 IP 网络场景的。由于 NDN 名字表与传统 IP 网络中查找表具有不同的结构以及特征，上述工作不能直接运用到 NDN 场景中。ndnSIM[11]是最近提出的一个基于 NDN 的仿真平台，但它侧重的是协议栈以及网络功能的仿真和验证，没有涉及关于流表、流量生成以及名字查找算法性能的比较。

针对以上问题，本文设计并实现了名字表以及流量生成工具，进而构建评价不同名字查找算法性能的平台 NDNBench。NDNBench 包含 4 个部分：种子表分析器、名字表产生器、名字流量产生器以及更新流量产生器。本文提出两种方法来产生名字表。对于给定的一个名字表，系统能够产生相应的包含查找和更新的合成流量。实验发现名字表和流量的特征将会在很大程度上影响 NDN 名字查找。系统将这些特征进行提取，形成了可以量化的参数并提供给用户。用户根据其需求通过调整这些参数，便可以生成具有不同特征的名字表和流量。本文对 NDN 场景下的更新行为进行了研究，以便用户产生合理的更新消息。

1 NDNBench 的工作流程

图 1 显示了 NDNBench 的工作流程图。本文从互联网上搜集了 10^7 个名字前缀来构成种子名字表。基于这个种子名字表，系统可以产生各种具有不同特征的名字表。其中，种子表分析器以种子表为输入，对影响其名字查找性能的特征进行概括，然后将这些特征信息以参数的形式进行量化。名字表产生器以这些参数为依据生成名字表。用户也可以根据自己需求修改这些参数。系统提供了两种方法合成名字表，用户可以选择其中的一种。类似地，名字流量产生器和更新流量产生器以名字表以及相应的参数集为输入，产生包含查找名字和更新信息的流量，用来冲击相应名字表。之后，系统将生成的 NDN 名字流量和更新消息进行混合，形成最终的合成流量。比较不同算法性能时，用户需要使用相同的名字表以及相应流量，可以对算法的吞吐率、内存占用以及更新性能等指标进行评价。用户也可以通过调整名字表和流量的相应参数来测试算法的可扩展性。

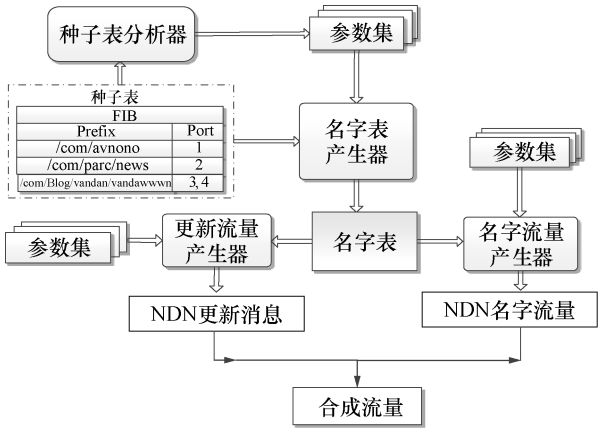

图 1 NDNBench 工作流程

参数实际上是对名字表以及流量特征的概括，它们将对名字查找算法的性能带来影响。因此，如何合理地提取参数，是系统生成名字表以及流量的关键，也是本文研究的重点。

2 NDNBench 功能模块设计

如图 1 所示，NDNBench 包含 4 个功能模块，各个模块之间通过相互配合生成名字表以及测试流量。本节将依次介绍包括种子表分析器、名字表产

生器、名字流量产生器以及更新流量产生器在内的各功能模块设计细节。

2.1　种子表分析器

该模块的作用是分析并抽取出名字表的关键特征，即影响名字查找算法性能的特征。该模块的输入是名字表。通过对名字表的分析，该模块提取相应参数值并将其输出到参数文件中。

2.1.1　数据准备

首先，本文对 NDN 名字表以及流量中的内容名字的结构进行介绍。NDN 名字采用层次化命名机制，由很多词元组成，相邻词元之间采用“/”作为分隔符，比如内容名字“/com/parc/newsroom/new.html”中，“/com/parc”是网站名字，“/newsroom/new.html”则定位到了服务器上所请求的内容。这种层次化名字结构使得名字前缀可以以词元为基本单位进行聚合，从而减少了名字表的表项规模。名字的层次结构也使得名字查询时需要进行最长前缀匹配，因此这对线速名字查找提出了挑战。

名字表每个表项由名字前缀和相应下一跳端口组成。图 1 中显示了一个包含了 3 个表项的名字表。路由器从到来的数据包中提取出内容名字，然后在名字表中进行最长前缀匹配来得到下一跳端口。系统通过以下步骤获得种子名字表：

1）域名收集。本文使用网络爬虫来获取域名和 URL，然后从 URL 中提取域名。为了获得尽可能多的域名，本文还从 www.namejet.com 网站下载部分域名。截止目前，系统已经获得没有重复的域名个数达到 8×10^7。

2）格式转换。系统将域名转换成 NDN 名字前缀的格式。比如，将“www.google.com”转换为“/com/google/www”。

3）下一跳端口生成。对于每个名字前缀，系统可以通过 DNS 解析，得到一个或者多个 IP 地址。这些 IP 地址将在一个从 www.ripe.net 得到的路由表上执行最长前缀匹配查找，并得到相应下一跳端口。

4）端口关联。系统将名字前缀和相应下一跳端口进行关联。至此，系统合成了一条名字表表项。

NDN 和传统 IP 网络虽然各自的转发机制不同，但是二者都是通过链路发现协议来生成相应的路由表。因此，本文的从 IP 路由表转换得到 NDN 名字表的过程是合理的。本文从 8×10^7 个备选表项中随机选取了 10^7 个表项组成种子表，下文的实验都是基于这个种子表进行的。

2.1.2　名字表参数提取

NDNBench 功能之一是生成名字表。首先需要确定名字表的哪些参数能够影响名字查找算法的性能。比如，不同名字表的平均前缀长度不同，可能对名字查找产生影响。另外，不同查找算法对名字表的某一参数的敏感度不同。基于词元的查找算法对前缀中词元的个数比较敏感，但不受前缀中某一位是字母还是数字的影响。使用具有相同特征的名字表是不同查找算法比较的基础。

本文将 NDN 名字查找算法分为两类：基于词元查找与基于字符查找。本文分别列出和这些算法性能相关的参数。基于词元的查找算法最小查找单元是词元，对以下参数敏感：前缀中词元个数、前缀中每一级词元长度。基于字符的查找算法最小查找单位是字符，对以下参数敏感：名字前缀字符个数、各种字符在每个位置出现的频率。本文对种子名字表的相关参数进行测量，测量结果展现在后面的实验部分中。

考虑到这些参数都是相互独立的，因此系统采用模块化设计。当一个影响查找性能的参数需要加入时，它将很容易整合到种子表分析器、参数集以及名字表产生器中，而不会对其他参数产生影响。

2.2　名字表产生器

名字表产生器以参数集以及要生成的名字表的大小为输入。用户可以使用从种子表学习得到的参数集，也可以根据自己需求，修改这些参数。该模块有两种方法来生成名字表。这两种方法生成的名字表继承了参数集的特征。

1）依据参数的要求，系统从种子名字表中选取指定数量的名字表项。这种方法生成的新表是种子表的子集。

2）依据参数产生新的表项，不需要种子表为输入。这种方法能够产生更多灵活的表项。

合成名字表最后一步是去除名字表中重叠表项。最简单的方法是把新生成的表项逐条和其他已生成表项进行比较。这种逐一比较的方法使得当表项数目大时，表项生成的时间特别长，时间复杂度为 $O(N)$，N 是表项数目。为了加速这个查重过程，系统构建词元 trie 树。词元 trie 树中每个节点代表一个名字前缀的词元。当产生新的前缀表项时，系

统将其插入到词元 trie 树中去检查该条表项是否已经存在。该方法的时间复杂度简化为$O(\log_k N)$，其中 k 为 trie 树节点平均子树个数。通过该方法，重复的表项在生成阶段就已经可以被消除了。

2.3 名字流量产生器

只使用名字表来评测查找算法时，系统只能比较算法存储占用。为了能够对算法的吞吐率、更新性能以及可扩展性等进行评测，系统需要用流量来冲击名字表。名字流量产生器可以产生针对某个名字表的名字请求。设计名字流量产生器的关键就在于找到名字流量中可以影响到名字查找性能的参数。除了待查找名字个数外，系统的设计中还包含以下参数：

1) 命中率。在路由查找中，存在一些名字由于不能匹配名字表的任何表项而被丢弃(有的路由器会将这些流量通过默认表项进行转发)。为了仿真这种场景，本文定义命中率 $\mathrm{HR}=\mathrm{Num}_{\mathrm{used}}/\mathrm{Num}_{\mathrm{total}}$，$\mathrm{Num}_{\mathrm{used}}$是成功匹配名字表的名字数目，$\mathrm{Num}_{\mathrm{total}}$是到来的名字请求总数。

2) 查找强度。最长前缀匹配查找中，不同名字将匹配不同长度的名字前缀。本文将查找流量匹配的表项的词元长度分布定义为查找强度。例如，对于图 1 中的名字表，如果采用基于 trie 树结构构造，全部匹配包含两个词元的前缀 P1 要比全部匹配包含 3 个词元的前缀 P3 有着更高的查找吞吐率。这是因为在 trie 树结构下，词元个数越多，查找所需要的访存次数就越多。

虽然名字查找是以数据包为粒度进行的，但不同数据包之间的一些参数也会对查找性能产生影响。本文首先定义 NDN 中流的概念。本文将流定义为包含相同内容名字的一组数据包。实现名字流量产生器还考虑了关于网络流的以下参数：

1) 新流到达速率。本文将其定义为每 s 到达的新流个数。

2) 流内包数。本文将其定义为流持续时间内到达的数据包的个数。目前系统实现了流内包数服从负指数分布。

3) 流内包到达间隔。本文将其定义为同一个流内的相邻到达数据包的时间间隔。

4) 流内包距离。本文将其定义为同一个流中相邻的两个数据包所间隔的不属于该流的数据包的个数。

在高速骨干网络中，流量的访问呈现局部性的特征，高速缓存结构便会缓存未来一段时间内最可能访问到的表项或者流量，从而减少数据包对主表的访问，起到加速查找的作用。在产生 NDN 流量时，本文使用流内包到达间隔来量化描述流量时间局部性特征，使用流内包距离来量化描述流量空间局部性特征。

2.4 更新流量产生器

本节首先对 NDN 更新形成原因及其行为进行分析，之后从更新行为特征中提取参数，用于实现更新流量产生器。

2.4.1 NDN 更新形成原因分析

通常更新消息包含添加、删除或者更新表项等操作。当一条更新指令到达时，路由表需要完成更新所需要的操作，直到更新指令被处理完，期间不能进行查找，这将对路由表查找的吞吐率造成影响。突发的更新到达甚至会使得查找队列等待数据包溢出，从而导致丢包。随着网络拓扑不断变化，路由策略动态性增强，路由器的更新变得越来越频繁，这一问题在 NDN 网络中变得更加严重。这是因为除了上述原因外，下面两种情形也会发生路由表的更新：1) 内容的发布以及删除；2) 内容提供者移动性的增强。因此，NDN 场景下的更新变得更加具有挑战性。

2.4.2 NDN 更新行为分析

考虑到 NDN 网络目前尚未大规模实际部署，本文基于当前互联网进行分析。本文对当前真实的 IP 路由表以及更新流量进行测量，得到以下结果：

1) 更新数据包通常不会匀速到达而会呈现突发特性。本文的测量结果显示，更新速率最高时能达到 3.5×10^4 条/s，而平均速率为 6 条/s。文[12]的研究工作也有类似的结果。因此，路由器在实现线速查找算法的同时，也要保证能够处理突发更新操作。

2) 大部分更新消息只会导致小部分表项频繁发生更新。本文的测量数据表明，只有 4% 的表项频繁受到更新数据包的影响，而其他表项保持相对稳定。

3) 本文使用 trie 树构建路由表，发现 86.7% 的更新集中在叶子节点。之所以产生这样的结果是因为 trie 树中叶子节点通常代表的是边缘网络，这些网络相对来说拓扑更容易发生改变。

本文将通过一个例子来说明移动性如何影响到名字表的更新。前缀 P1“/com/google/learning/network.pd”和前缀 P2“/com/google/learning/

compiler.pdf”能够聚合成为“/com/google/learning/”。层次化的名字结构使得名字前缀能够聚合，因此减少了名字表的规模。但是，当产生 P1 前缀的内容提供者移动到新的网络后，之前 P1 所在的路由器需要将 P1 删除，P1 与 P2 不能再聚合；另外，新的网络中的路由器也需要将新到来的 P1 添加到名字表中。可见，NDN 对移动性的支持加剧了名字表的更新。

2.4.3　更新流量产生器的实现

基于以上分析，本文设计了一个更新流量产生器来仿真更新行为。更新流量产生器还可以根据用户给定的参数，为输入的名字表生成相应的更新流量。用户可以基于以下参数来生成更新流量：

1）到达速率。到达速率指的是每 s 到达的更新包的数目。

2）更新包到达时间间隔分布。更新数据包到达时间可以符合不同概率分布。产生器实现了一些常用概率分布函数，包括正态分布、均匀分布以及 Poisson 分布等。用户可以选择其中某种分布。

3）更新强度。和查找强度类似，在基于 trie 树的名字查找中，更新叶子节点通常要比更新中间节点花费更多的时间，这是因为更新叶子节点需要更多次数的访存才能够定位相应节点。本文通过定义更新节点的访存次数来量化更新强度。

3　基于 NDNBench 的实验

本节将基于 NDNBench 对现有的一些 NDN 名字查找算法的查找速率、存储占用、可扩展性以及更新性能进行评估。

3.1　实验配置

本文在一台普通商用服务器上实现 NDNBench 平台，并且进行查找算法性能比较。该机器配置有两个 Xeon E5645 CPU，主频为 2.4 GHz；内存为 48 GB DDR3 ECC（1 333 MHz）；主板为 ASUS Z8PE-D12X（INTEL S5520）；机器的操作系统为 Linux，版本为 Fedora15（版本号为 2.6.41.9-1.fc15.x86_64）。

3.2　种子表以及流量特征分析

实验基于第 2 节中的种子名字表开展。本文对种子名字表特征进行分析，相关参数呈现在表 1 和 2 以及图 2 中。

表 1　种子名字表中前缀词元级数分布

前缀词元级数	2	3	4	5	6	≥7
出现频率/%	58.98	34.85	5.64	0.64	0.01	0.06

表 2　种子名字表中前缀字符数目分布

字符数目区间	1～9	10～15	16～19	20～29	≥30
出现频率/%	2.18	30.09	28.77	34.00	4.96

从表 1 可以看出，种子表中将近 94%的前缀有 2 级或者 3 级词元。表 2 显示超过 90%的前缀字符数目集中在 10～29。种子表中每一级词元字符数目分布如图 2 所示，可以看出每一级中，3 个字符的词元占到了 50%～70%。

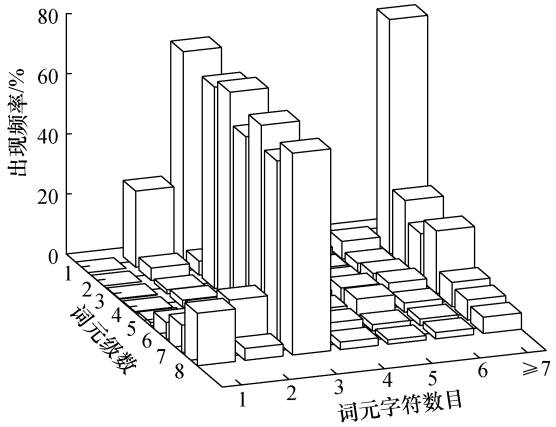

图 2　种子表中每级词元字符数目分布

名字流量包含一系列名字请求。这些名字通过连接名字表中的一条名字前缀和后缀库中的一条后缀组成。后缀是按照一定规则选择的。对于每个名字表，本文产生两种类型的名字流量：

1）平均负载流量。流量中的名字随机匹配名字表中的表项，即每个表项被访问到的概率是一样的。本文将这种类型的流量叫做平均负载流量。

2）高强度负载流量。流量中名字匹配名字表中前 10%的最长前缀。本文将这种类型的流量叫做高强度负载流量。

这两种类型的流量集有以下相同参数：待查找名字个数为 5×10^8，命中率为 100%。另外，为了测试算法的可扩展性，本文生成 10 个测试表，这些表的大小依次为 10^6，2×10^6，3×10^6，…，10^7，其他特征都和种子名字表一致，然后分别为其生成平均负载流量以及高强度负载流量。两种流量集的名字长度分布如图 3 所示。图 3 中，“a_1M.trace”表示冲击表大小为 10^6 的名字表的平均负载流量，而“w_1M.trace”则表示冲击表大小为 10^6 的名字表的高强度负载流量。

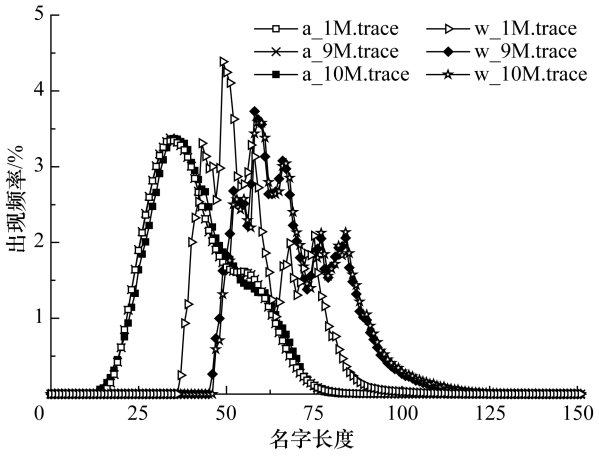

图 3 不同合成流量集中名字长度分布

强度相对不敏感。

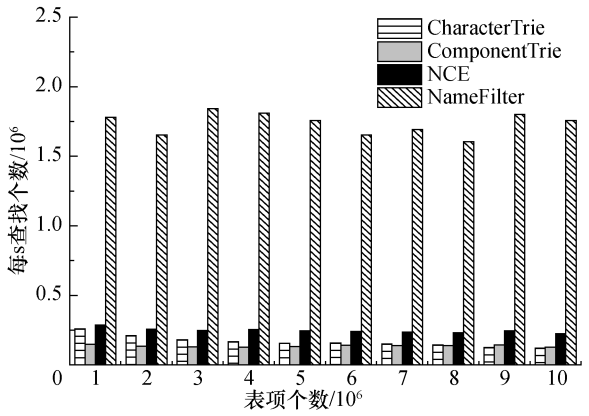

图 4 不同名字表规模下的名字表查找速度(平均负载)

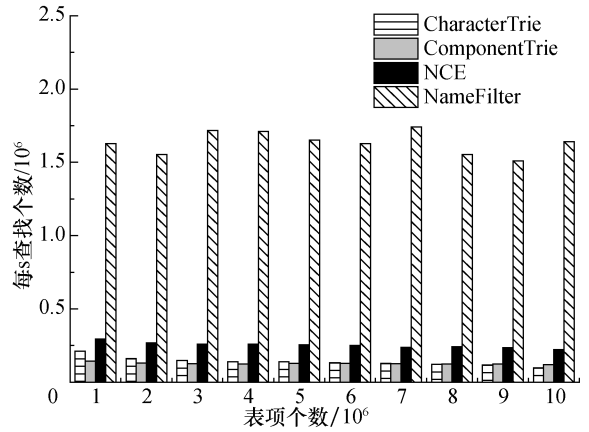

图 5 不同名字表规模下的名字表查找速度(高强度负载)

3.3 名字查找算法性能比较

本节利用 NDNBench 测试比较一些名字查找算法的性能。本文进行比较的算法包括基于字符查找树的方法(CharacterTrie)、基于词元查找树的方法(ComponentTrie)、NCE[3] 以及 NameFilter[4]。NCE 将名字词元进行编码，然后使用状态转移数组进行查找。NameFilter 是一种基于 Bloom 过滤器的名字查找方法：第一级使用面向字符的一次访存 Bloom 过滤器，第二级使用合并的 Bloom 过滤器。实验中，NameFilter 中误报率设置为$\leqslant 10^{-8}$，不同名字表对应的 Bloom 过滤器的大小也就可以计算出来。算法全部用 C++实现。

3.3.1 查找速度

图 4 显示了平均负载流量下，4 种查找方法的查找速度。可以看到，NameFilter 查找性能优于其他 3 种算法。特别地，对于种子表，NameFilter 仍能够到达到每 s 查找 1.76×10^6 个名字流量的查找速度，是 ComponentTrie 的 13.5 倍，CharacterTrie 的 14.7 倍，NCE 的 8 倍。另外，ComponentTrie、CharacterTrie 以及 NCE 这 3 种算法随着名字表表项数目的增加，查找速度呈下降趋势；而 NameFilter 查找性能则受表项数目影响不大，在查找性能方面显示出了良好的可扩展性。图 5 显示的是高强度负载流量下相应的查找速度测试结果。和平均负载流量下相比，除 NameFilter 外的其他 3 种算法在高强度负载流量下的查找速度均有所下降。这是因为对于这 3 种算法，匹配更长的前缀意味着访存次数的增加；而 NameFilter 中的两级结构和使用 Hash 函数的个数有关，和待匹配的前缀长度没有关系，也就是说 NameFilter 对参数查找

3.3.2 更新性能

为了评测不同查找算法的更新性能，本文采用更新时间作为指标，其具体定义为从名字表收到一条更新消息到更新操作执行完成之间的时间间隔，它反映了路由器对网络变化的敏感度。本文以插入性能为例评测 4 种查找算法。图 6 显示了 4 种算法

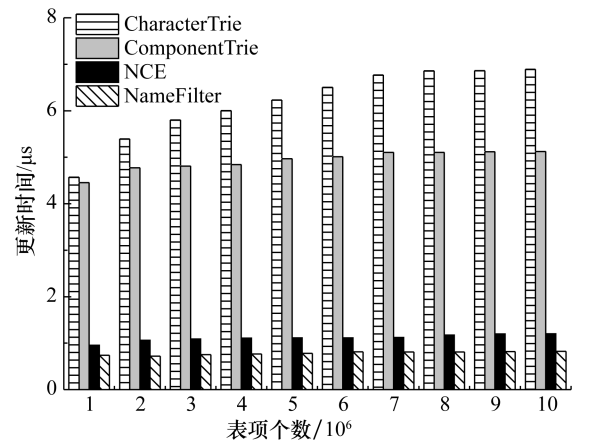

图 6 不同名字表规模下的插入性能

在不同名字表规模下的插入性能。可以看出，NCE 和 NameFilter 插入性能要优于其他两种算法，而且这两种算法的插入性能不随着路由表的规模变化而变化，显示出了良好的可扩展性。

3.3.3　存储占用

图 7 显示了 4 种算法的存储空间占用情况。可以看出，NameFilter 算法在各种名字表规模下的存储占用都要少于其他算法，显示该算法存储效果良好。另外，4 种算法的存储空间都随着名字表规模的增大而增大。NameFilter 曲线的斜率小于其他算法，显示出该算法良好的可扩展性，适用于更大规模的名字表。

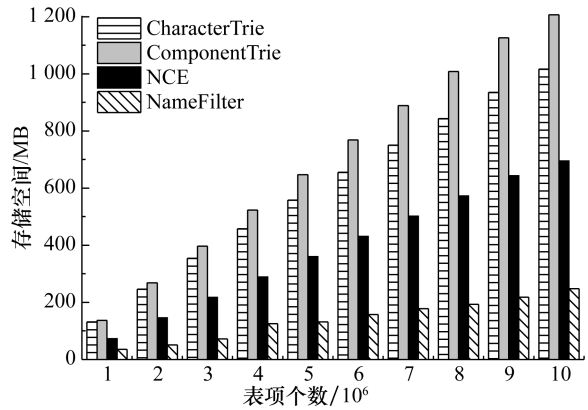

图 7　4 种算法在不同名字表规模下的存储空间

4　结　论

本文设计并实现了一个名字表以及流量生成工具 NDNBench，从而对 NDN 路由表、名字查找流量和更新流量等进行了模拟。该平台解决了目前尚未统一的 NDN 名字查找算法性能比较的问题。该平台已经运用在当前的一些名字查找算法的研究中[2-5]。下一步研究将从系统规模、性能方面对平台进一步优化。具体地，将生成更大规模的种子表来产生名字表以及流量，还将细化增加与删除更新对路由表性能的影响，分别单独考虑相应的流量模型，使得更新流量生成得更加合理。

参考文献　(References)

[1] ZHANG L, AFANASYEV A, BURKE J, et al. Named data networking [J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66 - 73.

[2] WANG Y, DAI H, JIANG J, et al. Parallel name lookup for named data networking [C]// 2011 IEEE Global Telecommunications Conference (GLOBECOM 2011). Houston, USA, 2011: 1 - 5.

[3] WANG Y, HE K, DAI H, et al. Scalable name lookup in NDN using effective name component encoding [C]// Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems. Washington DC, USA, 2012: 688 - 697.

[4] WANG Y, PAN T, MI Z, et al. NameFilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters [C]// 2013 Proceedings IEEE INFOCOM. Turin, Italy, 2013: 95 - 99.

[5] WANG Y, ZU Y, ZHANG T, et al. Wire speed name lookup: A GPU-based approach [C]// 2013 Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA, 2013: 199 - 212.

[6] TAYLOR D E, TURNER J S. ClassBench: A packet classification benchmark [C]//Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Miami, USA, 2005: 2068 - 2079.

[7] GANEGEDARA T, JIANG W, PRASANNA V. FRuG: A benchmark for packet forwarding in future networks [C]// 29th International Performance Computing and Communications Conference. Albuquerque, USA, 2010: 231 - 238.

[8] CASTELINO M, GUNTURI R, FILAURO V, et al. Benchmarks for IP forwarding tables [C]// IEEE International Conference on Performance, Computing, and Communications. Phoenix, USA, 2004: 123 - 130.

[9] WANG M, DEERING S, HAIN T, et al. Non-random generator for IPv6 tables [C]// Proceedings of 12th Annual IEEE Symposium on High Performance Interconnects. Stanford, USA, 2004: 35 - 40.

[10] ZHENG K, LIU B. V6Gene: A scalable IPv6 prefix generator for route lookup algorithm benchmark [C]// International Conference on Advanced Information Networking and Applications. Vienna, Austria, 2006: 6 - 12.

[11] MASTORAKIS S, AFANASYEV A, MOISEENKO I, et al. ndnSIM 2.0: A new version of the NDN simulator for NS-3, NDN-0028 [R]. Los Angeles, USA: University of California-Los Angeles, 2015.

[12] YANG T, MI Z, DUAN R, et al. An ultra-fast universal incremental update algorithm for trie-based routing lookup [C]// 2012 20th IEEE International Conference on Network Protocols (ICNP). Austin, USA, 2012: 1 - 10.

（责任编辑　李丽）