

## IPv4 网络与 IPv6 互联网的无状态通信机制

韩国梁\*, 盛茂家, 包丛笑, 李 星

(清华大学 网络科学与网络空间研究院, 北京 100084)

(\*通信作者电子邮箱 guoliang, taurus@gmail.com)

**摘 要:**在 IPv4/IPv6 过渡进程中, 针对一些尚未升级到 IPv6 的 IPv4 网络仍需要与 IPv6 互联网互联互通的问题, 提出了一种无状态的双向通信机制, 完善了目前 IPv4/IPv6 翻译的整体框架。首先, 针对 IPv4 服务器被 IPv6 用户访问的场景和 IPv4 网络访问 IPv6 资源的场景分别提出了相应的通信流程, 结合已有的无状态通信机制, 形成统一的 IPv4/IPv6 无状态通信框架。其次, 对 IPv6 到 IPv4 的单向映射函数的需求进行了分析, 提出了三个定量评价标准, 并用实际数据对各种哈希函数进行了分析和比较。实验表明, FarmHash 哈希算法的处理时间短、冲突频率低、反向查询复杂度低, 适用于 IPv4 网络与 IPv6 互联网相互通信的两类场景, 从而验证了该机制的可行性; 与现有的有状态通信机制相比, 该机制具有很好的可扩展性和可溯源性, 能够支持双向发起的通信, 因此可以有效促进 IPv4 到 IPv6 的过渡。

**关键词:**IPv4/IPv6 过渡; 无状态翻译; 哈希函数; 可扩展性; 可溯源性

**中图分类号:** TP393.071 **文献标志码:** A

### Stateless communication mechanism between an IPv4 network and the IPv6 Internet

HAN Guoliang\*, SHENG Maojia, BAO Congxiao, LI Xing

(Institute of Network Science and Cyberspace, Tsinghua University, Beijing 100084, China)

**Abstract:** In the IPv4/IPv6 transition process, since some legacy IPv4 networks still need to communicate with the IPv6 Internet, the stateless communication mechanism between an IPv4 network and the IPv6 Internet, which complements the current IPv4/IPv6 translation framework, was proposed. First, the communication procedures in two related scenarios were demonstrated. The two scenarios include IPv6 Internet clients accessing IPv4 servers and IPv4 clients accessing IPv6 Internet servers. The one-way IPv6-IPv4 address mapping function is the key component of the mechanism. Therefore, the requirements and three quantitative criteria of the one-way mapping function were discussed. Afterwards, multiple Hash functions as the candidates of the one-way mapping function were compared and analyzed with the real user data of large IPv6 websites and real IPv6 server addresses. The simulation results show that the FarmCity Hash function is suitable to be deployed in the above two scenarios because it has short average processing time, low collision rate and low reverse query complexity. It also verifies the validity of the stateless communication mechanism. Compared with current stateful communication mechanisms, the stateless mechanism has better scalability and traceability. Moreover, the capacity for bidirectional communication facilitates a smooth migration path towards the IPv6 Internet.

**Key words:** IPv4/IPv6 transition; stateless translation; Hash function; scalability; traceability

## 0 引言

由于全球 IPv4 地址的耗尽, 由 IPv4 向 IPv6 的过渡势在必行, 但 IPv6 从设计之初便与 IPv4 并不兼容。在众多的 IPv4/IPv6 过渡技术中, 无状态 IPv4/IPv6 翻译技术<sup>[1-3]</sup>使得新建的 IPv6 网络能与现有的 IPv4 互联网双向互联互通, 因此能够有效地促进 IPv4 到 IPv6 的过渡进程。同时, 由于在无状态翻译场景中, IPv4 和 IPv6 地址之间的映射基于固定算法<sup>[2]</sup>, 因此互联网服务提供商 (Internet Service Provider, ISP) 可以对 IPv4 用户和 IPv6 用户进行统一管理, 包括流量管理、用户过滤、有效溯源等操作<sup>[4]</sup>。

在 IPv4 到 IPv6 的过渡过程中, 仍有一些 IPv4 网络尚需要时间过渡到 IPv6, 但仍需要和 IPv6 互联网互联互通。比如

一些信息资源提供商 (Information Content Provider, ICP) 需要被 IPv6 互联网的用户访问, 或者一些 IPv4 的接入网希望能访问到 IPv6 互联网上的资源。现有的技术并不能很好地解决这个问题。NAT-PT (Network Address Translation-Protocol Translation)<sup>[5]</sup>能解决纯 IPv4 节点和纯 IPv6 节点相互通信的问题, 但由于其不可扩展性和复杂性已经被国际互联网工程任务组 (Internet Engineering Task Force, IETF) 废弃<sup>[6]</sup>。NAT64 有状态翻译技术<sup>[7-8]</sup>能解决 IPv6 互联网访问 IPv4 网络的问题, 但是需要边缘设备为每个会话动态分配一个 IPv4 地址和端口, 并保存和跟踪相应的会话状态。当有大量的用户访问时, 翻译网关需要对大量会话粒度的状态作更新和同步, 并需要作详细的日志记录。这些复杂的操作会影响到该翻译网关的性能, 容易形成单点故障并降低网络应用的用户

收稿日期: 2015-03-30; 修回日期: 2015-06-08。 基金项目: 国家科技支撑计划项目 (2012BAH01B02)。

作者简介: 韩国梁 (1987-), 男, 山西太原人, 博士研究生, 主要研究方向: 计算机网络; 盛茂家 (1992-), 男, 云南昭通人, 硕士研究生, 主要研究方向: 计算机网络; 包丛笑 (1967-), 女, 上海人, 副研究员, 主要研究方向: 计算机网络、多媒体通信系统; 李星 (1956-), 男, 陕西蒲城人, 教授, 博士生导师, 主要研究方向: 信号与信息处理、计算机网络、多媒体通信系统。

体验。同时,由于 IPv4 网络中的服务器记录的 IPv4 源地址都是动态生成的,在服务端想要区分用户或对用户作溯源就变得非常困难。而 NAT64 技术不能解决 IPv4 网络访问 IPv6 互联网的问题。

针对以上问题,本文提出了无状态翻译的解决方案,可以使 IPv4 网络和 IPv6 互联网能够互联互通。与现有的有状态通信机制相比,该机制具有很好的可扩展性和可溯源性,能够支持双向发起的通信,因此可以有效促进 IPv4 到 IPv6 的过渡。

## 1 无状态 IPv4/IPv6 通信框架

无状态通信的核心在于 IPv4 报文头部和 IPv6 报文头部的各字段之间能够建立简单的基于算法的映射关系,这样翻译网关无需保存和跟踪动态的映射关系,仅根据报文中包含的信息就可进行翻译。RFC 6145<sup>[3]</sup> 规定了无状态 IPv4/IPv6 协议翻译的规则。在翻译过程中,IPv4/IPv6 无状态地址映射是最困难之处,因为 IPv4 的 32 位地址空间和 IPv6 的 128 位地址空间相差很大,不可能将 IPv6 地址空间完全映射到 IPv4 地址空间中去,必须根据实际的需求设计相应的映射规则。

### 1.1 IPv6 网络与 IPv4 互联网无状态互联互通

RFC 6052<sup>[2]</sup> 规定了 IPv6 网络与 IPv4 互联网相互通信时 IPv4/IPv6 地址映射的规则,如图 1(a) 所示。整个 IPv4 地址空间 ( $G_4$ ) 可以无状态一一映射到一个 IPv6 子网 ( $IVIC_6$ ) 中,映射规则为将 IPv4 地址嵌入 IPv6 前缀后面,其余位补零(映射 1),这样的 IPv6 地址称为 IPv4 转换地址 (IPv4-converted address),不被真实的 IPv6 计算机使用。为使 IPv6 网络的计算机地址能无状态映射到 IPv4 地址空间,ISP 为它们分配特殊的 IPv6 地址 (IVI6),如图 1(a) 中右边阴影部分所示,称为 IPv4 可译地址 (IPv4-translatable address)。IVI6 地址集合可以和 IPv4 地址空间中的一小段子网 (甚至一个 IPv4 地址) 建立一对一或多对一的无状态映射关系 (映射 2) (为节省 IPv4 地址,一个 IPv4 地址可以根据端口的静态划分无状态映射到多个 IPv4 可译地址)。称 IVI6 地址在 IPv4 地址空间中的像为 IVI4 地址,不被真实的 IPv4 计算机使用。

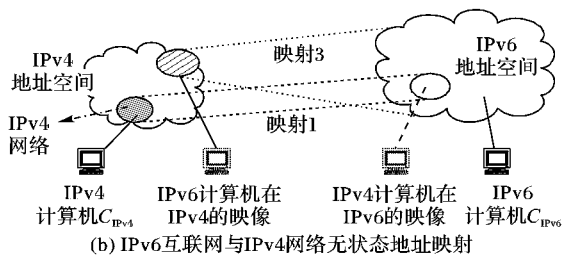
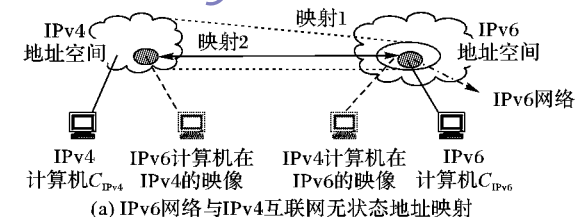


图1 无状态 IPv4/IPv6 地址映射模型

根据 RFC 6052 定义地址映射规则,拥有 IVI6 地址的 IPv6 计算机  $C_{IPv6}$  可以和 IPv4 互联网中的任一台计算机  $C_{IPv4}$  通信,它认为它的通信对端就是  $C_{IPv4}$  在 IPv6 中的像;而  $C_{IPv4}$  也可以通过访问  $C_{IPv6}$  在 IPv4 中对应的 IVI4 地址访问到  $C_{IPv6}$ ,它认为通信对端就是该 IVI4 地址。由此实现了 IPv6 网

络和 IPv4 互联网双向透明的无状态通信。从 ISP 角度来讲,由于新建 IPv6 网络可以同时与 IPv4 和 IPv6 互联网互联互通,因此 ISP 可以部署纯 IPv6 接入网,仍为其用户提供 IPv4 和 IPv6 互联网的访问服务;从网络内容服务商 (ICP) 角度来讲,其网络可以升级为纯 IPv6,仍可以为 IPv4 和 IPv6 互联网中的用户提供访问服务。因此,IPv6 网络与 IPv4 互联网无状态互通的模式可以有效促进 IPv4 到 IPv6 的过渡进程。

### 1.2 IPv4 网络与 IPv6 互联网无状态互联互通

依照以上地址映射的思路,如果 IPv4 网络需要和 IPv6 互联网互联互通,需要两个无状态映射,分别将 IPv4 网络的地址映射到 IPv6 地址空间,以及将 IPv6 互联网的地址映射到 IPv4 的地址空间。第一个映射是容易的,用 RFC 6052 中的映射 1 便可完成;而第二个映射是困难的,因为如前所述,IPv6 的地址空间比 IPv4 地址空间要大很多。但是,需要和某个 IPv4 网络相互通信的 IPv6 互联网用户或资源要比 IPv6 地址空间小得多。根据最新的统计报告,截止 2015 年 3 月,全球的 IPv6 用户数量仅有 1.02 亿<sup>[9]</sup>,仅占到全球 30 亿互联网人口的 3.4%。而经笔者分析(具体见第 3 章),中国教育与科研计算机网站 (<http://www.edu.cn>) 2014 年独立 IPv6 访问量为 270 万,单月的峰值 IPv6 访问量为 33.4 万,甚至远小于一个 A 类 IPv4 地址的地址空间。另一方面,根据 Alexa 的测量<sup>[10]</sup>,在最流行的 100 万网站中,支持 IPv6 的网站仅有 5.9 万,占比不到 6%。因此,在 IPv4 向 IPv6 过渡的初期,可能与某个特定 IPv4 网络相互通信的 IPv6 地址集合会远小于 IPv4 的地址空间。此时,可设计单向函数作无状态 IPv6-IPv4 地址映射(映射 3),将任意的 IPv6 地址映射到 IPv4 地址空间的一个子集中(如图 1(b) 中影线阴影所示),并尽量减少映射冲突,从而实现 IPv4 网络与 IPv6 互联网的无状态通信。

IPv4 网络与 IPv6 互联网的通信具体可分为两种应用场景,分别如下所述。

#### 1.2.1 场景一:IPv6 互联网访问 IPv4 网络

此场景通常用于 IPv4 ICP 网络仍需要时间过渡到 IPv6,但需要被 IPv6 互联网中的用户访问的情况。如图 2(a) 所示,ICP 网络首先为其每台 IPv4 服务器注册两条域名系统 (Domain Name System, DNS) 记录,A 记录为其本身的 IPv4 地址,AAAA 记录为根据无状态映射 1 合成的 IPv6 地址。IPv6 互联网用户访问 IPv4 服务器的步骤<sup>[11]</sup> 如下所示:

1) IPv6 用户请求目标服务器的 IPv6 地址,发送 IPv6 报文。

2) 无状态翻译器翻译 IPv6 报文。目标 IPv6 地址被无状态映射回 IPv4 地址(映射 1),源 IPv6 地址被单向函数映射为 IPv4 地址(映射 3)并将该映射关系保存在静态映射表中。如果有冲突,更改源端口并将更改前和更改后的端口值保存在映射表中。翻译后的 IPv4 报文被发送到 IPv4 服务器。

3) IPv4 服务器返回 IPv4 报文。

4) 无状态翻译器翻译 IPv4 报文。源 IPv4 地址被无状态映射到 IPv6 地址(映射 1),目标 IPv4 地址通过查询静态映射表映射为相应 IPv6 地址。如果该 IPv4 对应的某一映射表项包含了端口更改,且该表项中更改后的端口值与目标端口值相等,需将目标端口值复原。

5) 后续报文直接查表,无需使用上述单向映射函数。

如步骤 2) 和 4) 所示,使用端口移位的方式处理冲突,即

将发生冲突的若干 IPv6 地址存储在一个链表中(链接法),并更改其源端口保证不同会话映射后的五元组依旧不同。这就要求单向映射函数能达到好的反向查询性能,即反向查询某 IPv4 地址的期望时间复杂度最小。

对 ICP 中的 IPv4 服务器来说,如果单向映射函数的冲突足够少,其日志记录的每个 IPv4 用户地址都可以通过静态映射表找到源 IPv6 地址,因而实现了较好的溯源性和安全性。同时静态的映射也方便网络管理员作用户过滤、流量管理等操作。静态映射表可以存储在和翻译器有高速链路的带外数据库或内存中。

### 1.2.2 场景二:IPv4 网络访问 IPv6 互联网

此场景可用于 ICP 的 IPv4 服务器访问 IPv6 互联网,或者尚未升级到 IPv4 的接入网络访问 IPv6 互联网,或其他类似的应用情况。如图 2(b)所示,IPv4 网络用户访问 IPv6 互联网服务器的步骤如下所示:

- 1) IPv4 用户请求目标服务器的 IPv6 地址。
- 2) DNS 查询服务器的 AAAA 记录,使用单向函数映射为 IPv4 地址(映射 3)并返回给 IPv4 用户。映射完成后,在静态映射表中记录映射关系,如有冲突作二次映射并记录,也可根据用户经常访问的网站事先制作为静态彩虹表。
- 3) IPv4 用户发送 IPv4 报文。
- 4) 无状态翻译器翻译 IPv4 报文并发送至 IPv6 服务器。源 IPv4 地址被无状态映射到 IPv6 地址(映射 1),目标 IPv4 地址通过查询静态映射表映射为相应 IPv6 地址。
- 5) IPv6 服务器返回 IPv6 报文。
- 6) 无状态翻译器翻译 IPv6 报文并发送回 IPv4 用户。目标 IPv6 地址被无状态映射回 IPv4 地址(映射 1),源 IPv6 地址通过查询静态映射表映射为 IPv4 地址。
- 7) 后续报文直接查表,无需使用上述单向映射函数。

场景二与场景一的不同之处在于 DNS 作单向映射函数时,仅能查询地址,不能对端口进行修改。但是,只要用户访问的 IPv6 目标地址在已知范围内,因此总可以设计完美哈希函数为这些目标地址事先建立好映射关系表,并存储在和翻译器与 DNS 均有高速链路的数据库或内存中。

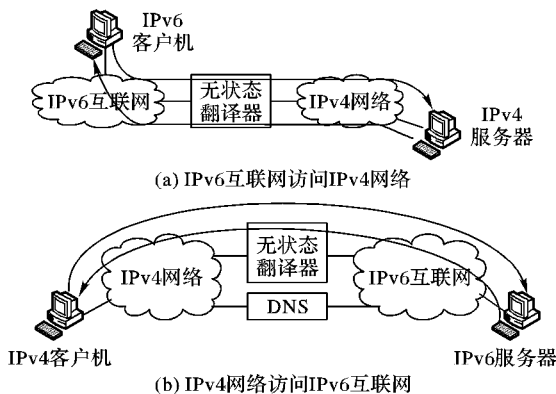


图2 IPv4 网络与 IPv6 互联网通信的两种应用场景

综上,本文设计了 IPv4 网络和 IPv6 互联网之间的双向无状态通信机制,通过引入 IPv6 地址到 IPv4 地址的单向哈希函数,使得解决方案具有良好的可溯源性和网络运维性。与现有的无状态 IPv4/IPv6 翻译机制相结合,对处于 IPv6 过渡期的大型 ICP 来说,无论服务器是 IPv4 还是 IPv6,都能无状态地为 IPv4 互联网和 IPv6 互联网提供服务。

该机制的关键是设计一个好的 IPv6-IPv4 地址单向映射函数,下面将对该函数的设计需求和评价标准作简单介绍,并针对需求设计合适的映射函数。

## 2 IPv6-IPv4 地址单向映射函数设计

### 2.1 设计需求

IPv6-IPv4 单向映射函数将 128 位的任意 IPv6 地址映射到 IPv4 地址空间的某个子集  $S$  中,如图 1(b)中影线阴影部分所示。在上述两个场景中,由于 IPv4 网络通常还要与 IPv4 互联网通信,因此  $S$  应不包含服务器所在的 IPv4 子网以及所有公网 IPv4 地址。如果服务器未使用私有 IPv4 地址,则  $S$  可包括私有的地址块和互联网数字分配机构(Internet Assigned Numbers Authority, IANA)保留的地址块<sup>[12]</sup>(这些地址不会进入全球路由表)。针对不同的应用场景,  $S$  应调整至  $|S|$  达到最大。一般情况下,10.0.0.0/24 私有地址块和 IANA 预留的 IPv4 地址块足够提供  $2^{26}$  大小的值域范围。

不失一般性,本文仅考虑  $S$  仅包含一个 IPv4 子网的情况。事实上,如果  $S$  包括多个 IPv4 子网,可以建立一个虚拟的子网  $S_1$ ,其大小为  $|S|$ ,且与  $S$  之间建立简单的一一映射的关系。这样将 IPv6 地址空间映射到  $S_1$  就等价于映射到  $S$ 。

在场景一中,一些服务器需要在映射时对用户的源 IPv6 地址保持前缀,即有特定长度公共前缀的 IPv6 地址能映射到同一个 IPv4 子网中,以便更精细的用户管理。假设单向映射函数为  $f$ , IPv6 地址空间为  $G_6$ , 希望将公共前缀长度至少为  $l_6$  的 IPv6 地址映射到一个前缀长度至少为  $l_4$  的 IPv4 子网中去,则该需求可表示为:

$$\forall A_6^1, A_6^2 \in G_6, A_6^1 \vee A_6^2 \geq l_6 \Leftrightarrow f(A_6^1) \vee f(A_6^2) \geq l_4$$

### 2.2 单向映射函数选择与设计

目前研究较为成熟的哈希算法可直接用作本文中的单向映射函数。常用的哈希函数包括 MD5、SHA-1、SHA-256、SHA-512、SHA-384、SHA224(加密哈希函数<sup>[13]</sup>)、RS、PJW、ELF、DJB、BP、FNV<sup>[14]</sup>、CityHash<sup>[15]</sup>、FarmHash<sup>[16]</sup>(非加密哈希函数)等。其中:非加密哈希函数通常用于字符串压缩,只要指定源字符串的字节数为 16,便可用于 IPv6 地址的映射;而加密哈希函数(MD5 或 SHA 家族)通常将 512 比特(64 字节)或 1024 比特(128 字节)视为一个区块进行计算,在映射 IPv6 地址时,处理时间可能会相对较长。

根据值域  $|S|$  的大小不同,需要对上述哈希函数的输出作截位处理。只要哈希函数的输出分布足够均匀,从任何位置截位都可以得到均匀分布。

在场景一中,为满足保持前缀的映射需求,本文使用分段哈希函数。假设值域  $|S|$  是一个前缀长度为  $l$  的子网, IPv6 地址  $A_6$  通过分段哈希函数映射为  $A_4$ ,则有:

$$\begin{cases} f_1(A_6[l_6, l_6]) = A_4[l+1, l+l_4] \\ f_2(A_6[l_6+1, 128]) = A_4[l+l_4+1, 32] \end{cases}$$

如第 3 章所示,选择不同的  $l_6$  和  $l_4$  会影响到哈希函数的性能。因此服务器应谨慎选择以上参数,以便在保持前缀和冲突率两者之间取得合适的折中方案。

在场景二中,如果目标 IPv6 服务器数量较少,可直接使用上述哈希函数制作静态映射表。当上述哈希函数的冲突率较高时,只要 IPv6 服务器数量小于 IPv4 子网大小,可以通过



设计完美哈希函数<sup>[17]</sup>提供冲突率为零的静态映射。

### 2.3 评价标准

评价 IPv6-IPv4 地址单向映射函数的指标包括处理时间和冲突频率。在具体的应用场景中,管理员应选择处理时间短且冲突频率低的单向映射函数。对于场景一的单向映射函数,评价指标还包括反向查询复杂度。为提高映射操作的效率,管理员也应选择反向查询复杂度较低的单向映射函数。

处理时间表示单向映射函数处理一个 IPv6 地址平均所用时间。处理时间的绝对值与运算平台相关,在同一平台下,处理时间越短的单向函数越好。

冲突频率表示对于一组特定的 IPv6 地址集,单向映射函数生成的 IPv4 地址的重复频率。冲突频率越小的单向函数越好。假设 IPv6 地址集有  $N$  个 IPv6 地址,值域  $S$  包括  $M$  个 IPv4 地址,其中  $N > M$ 。 $N_i$  表示映射到第  $i$  个 IPv4 地址的 IPv6 地址个数,若  $N_i > 1$  则冲突个数为  $N_i - 1$ 。冲突频率可表示为:

$$p = \frac{1}{N} \sum_{i=1}^M (N_i - \varepsilon(N_i - 1)) = 1 - \frac{1}{N} \sum_{i=1}^M \varepsilon(N_i - 1)$$

其中  $\varepsilon(x)$  为阶跃函数:

$$\varepsilon(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

显然,要使冲突频率最小,需使  $N_i > 0$  的 IPv4 地址数量最多,即  $N$  个 IPv6 地址映射到不同的  $N$  个 IPv4 地址。

场景一中的反向查询复杂度表示将一组 IPv6 地址映射至 IPv4 地址集合  $S_0$  后,由  $S_0$  中的 IPv4 地址反查 IPv6 地址的期望时间复杂度。期望时间复杂度越低的单向函数越好。由于相同 IPv4 地址通过端口移位处理冲突,因此反查某个 IPv4 地址时,需要遍历其对应的所有 IPv6 地址,直到匹配到某个移位后端口为止。假设每个 IPv6 地址出现的概率均为  $N^{-1}$ ,则第  $i$  个 IPv4 地址需要查询的期望次数为:

$$E(T_i) = \begin{cases} \frac{1}{N_i} \left( N_i + \sum_{j=2}^{N_i} j \right) = \frac{N_i^2 + 3N_i - 2}{2N_i}, & N_i \geq 2 \\ 1, & N_i = 1 \\ 0, & N_i = 0 \end{cases}$$

总体的期望查询次数为:

$$\begin{aligned} E(T) &= \frac{1}{N} \sum_{i=1}^M N_i E(T_i) = \\ &= \frac{1}{2N} \sum_{i=1}^M (N_i^2 + 3N_i - 2) \varepsilon(N_i - 1) = \\ &= \frac{1}{2N} \sum_{i=1}^M N_i^2 - \frac{1}{N} \sum_{i=1}^M \varepsilon(N_i - 1) + \frac{3}{2} = \\ &= \frac{1}{2N} \sum_{i=1}^M N_i^2 - (1 - p) + \frac{3}{2} = \\ &= \frac{1}{2N} \left( \sum_{i=1}^M \left( N_i - \frac{N}{M} \right)^2 + \frac{N^2}{M} \right) + p + \frac{1}{2} = \\ &= \frac{M}{2N} \sigma^2 + p + \frac{N + M}{2M} \end{aligned}$$

因此,期望查询次数与  $N_i$  的方差  $\sigma^2$  及冲突频率  $p$  成线性正相关。由于两者都在一一映射时达到最小值,代入可得总体期望查询次数最小值为 1。在冲突频率相同的基础上,  $\sigma^2$  越大,期望查询次数越多,因此该指标也反映出映射函数的平衡性。

## 3 实验验证

本文采集了中国教育与科研计算机网站 2014 年的访问日志数据,提取出 1 个月、3 个月、6 个月、9 个月和 12 个月的独立 IPv6 地址集合,集合大小分别为 14 万、48 万、125 万、182 万和 270 万。在此数据基础上,分析比较了在 IPv6 互联网访问 IPv4 网络场景(场景一)中,2.2 节提到的常用哈希函数的性能。针对场景二,本文采集了目前流行的 19 万个 IPv6 网站,共解析出 4.8 万个可访问的 IPv6 服务器地址,并分析了不同哈希函数的效果。

### 3.1 处理时间比较

本文的实验环境为 Inter Core i7 2.7 GHz CPU, 4 GB 1333 MHz 内存和 MAC OS Yosemite 64 位操作系统。针对上述 IPv6 地址集合,计算各哈希函数映射一个 IPv6 地址的平均处理时间,如图 3 所示。

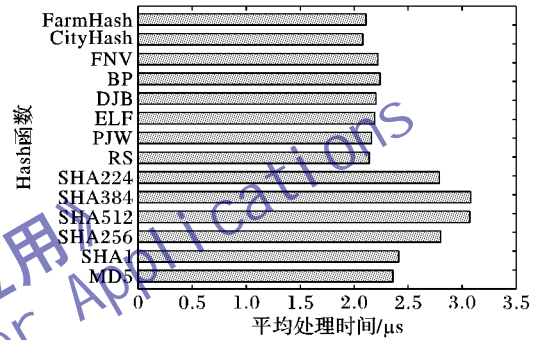


图3 不同哈希算法处理时间比较图

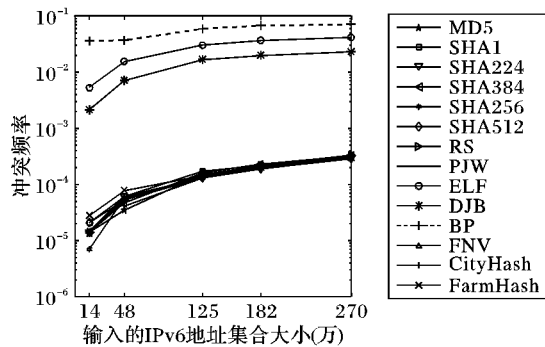
可以看出,加密函数的处理时间通常比非加密函数处理时间要长一些。在加密函数中,处理时间较短的有 MD5 和 SHA1,而本文列举的非加密哈希函数处理时间均在 2~2.2 μs,较加密哈希函数有明显的优势。

### 3.2 冲突频率比较

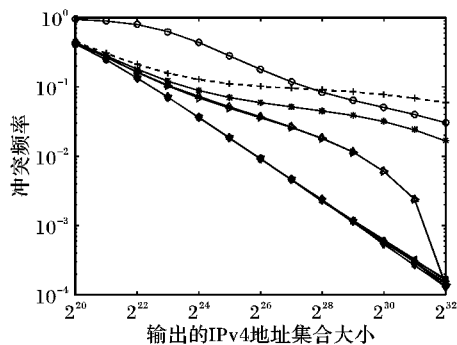
首先设置哈希函数的值域是整个 32 位 IPv4 地址空间,比较各种哈希函数的冲突频率  $p$  与 IPv6 集合大小  $N$  之间的关系,如图 4(a)所示;接下来固定  $N$  为 125 万(6 个月 IPv6 地址数量),修改输出值域的大小  $M$ ,比较各种哈希函数的冲突频率  $p$  与  $M$  之间的关系,如图 4(b)所示。

可以看到,当  $M$  不变时,冲突频率随着  $N$  的增长而变大;当  $N$  不变时,冲突频率随着  $M$  的增长而变小。该结果与泊松装填模型<sup>[18]</sup>相吻合,即冲突概率与装载因子 ( $N/M$ ) 成反相关关系。相比其他哈希函数,BP、ELF、PJW 和 DJB 四个哈希函数的冲突概率明显大很多。在值域为 32 位 IPv4 地址空间条件下,输入 48 万个 IPv6 地址集合时,绝大部分哈希函数的冲突频率均小于万分之一;当输入 125 万个 IPv6 地址时,绝大部分哈希函数的冲突频率均小于万分之二。在值域为 26 位 IPv4 地址空间条件下(10.0.0.0/24 和 IANA 预留的 IPv4 地址块足够提供  $2^{26}$  大小的值域),输入 125 万个 IPv6 地址时,绝大部分哈希函数的冲突概率小于百分之一。也就是说,在 IPv4 到 IPv6 过渡初期,一个 IPv6 独立用户数量在 125 万以下的 IPv4 网络均可使用本文所述机制为 IPv6 互联网提供无状态访问服务,且冲突概率非常小。当 IPv6 用户逐渐增多时(达到几百万或几千万的量级),哈希函数的冲突率会逐渐增加,这时管理员可每隔一段时间(如一个月)更新一次静态

映射表,根据实验结果,冲突率仍可以降到万分之一以下。当 IPv6 用户继续增加时,管理员应考虑尽快将服务器升级到纯 IPv6,使用 1.1 节所述机制为 IPv4/IPv6 互联网提供服务。



(a) 冲突频率与输入IPv6集合大小的关系



(b) 冲突频率与输出的IPv4地址集合大小的关系

图4 不同哈希算法冲突频率比较

### 3.3 反向查询复杂度比较

本文还比较了各种哈希函数的反向查询复杂度与各 IPv6 地址集合大小之间的关系,如图 5 所示。可以看出,对同一个哈希函数,反向查询复杂度和 IPv6 地址集合大小成正相关关系。除 BP、ELF、PJW(与 ELF 的数据相同)和 DJB 四种函数外,其他哈希函数均可以实现较小的反向查询复杂度。因此,本文所述机制可以实现较快的反查性能。

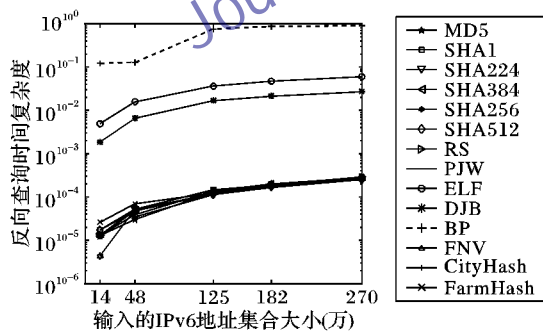


图5 不同哈希算法反向查询复杂度比较

### 3.4 设计冲突率低的分段哈希函数

针对场景中保持前缀的需求,对分段哈希函数进行仿真实验,值域为 32 位 IPv4 地址空间。对不同的  $l_6$  和  $l_4$  取值,使用性能较好的 FarmHash 函数分别对 IPv6 地址的两段进行映射。在  $N$  为 125 万(6 个月 IPv6 地址数量)条件下,不同  $l_6$  和  $l_4$  取值对冲突频率的影响如图 6 所示。

可以看出,当  $l_6$  等于 48 位时,  $l_4$  大于 5 位就会使冲突频率迅速增长,说明 IPv6 用户地址在前 48 位的分布均匀性要比后 80 位差很多。这是由于目前 IPv6 前缀数量很少,仅用少数几个比特来表示便可以,而后 80 位尤其是后 64 位通常包

含了用户的 MAC 地址信息或者随机生成的标识信息,信息量巨大,因此需要占用很多比特来表示。当  $l_6$  大于 96 位后,  $l_4$  也应增加到 20 位才能保持较低的冲突率。因此,管理员应考虑到 IPv6 地址的分布特征,合理选择  $l_4$  和  $l_6$ ,以便在保持单向映射的无状态特性的同时更好地对用户进行管理。

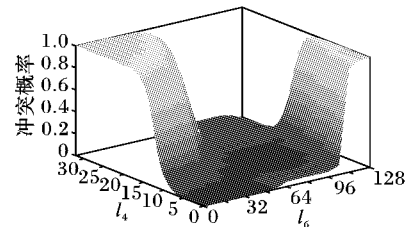


图6 不同前缀反向查询复杂度比较

### 3.5 场景二的冲突频率

本文采集了目前流行的 19 万个 IPv6 网站,共解析出 4.8 万个可访问的 IPv6 服务器地址,使用上述各哈希函数对这些 IPv6 地址进行单向映射,比较其各自的冲突频率。实验表明,当值域为 32 位 IPv4 地址空间时,除 BP、ELF、PJW 和 DJB 外的哈希函数至多发生一次冲突;而当值域空间为  $2^{26}$  时,如表 1 所示,除 BP、ELF、PJW 和 DJB 外的哈希函数的冲突数均不超过 25 个,其中 FarmHash 函数仅发生 8 次冲突,冲突率仅 0.016%。实验表明,本文所述无状态机制也可很好地适用于场景二。

表1 场景二中各哈希函数的冲突数量

(值域为  $2^{26}$ , 样本数量 4.8 万)

| 哈希函数   | 冲突数 | 哈希函数 | 冲突数    | 哈希函数     | 冲突数    |
|--------|-----|------|--------|----------|--------|
| SHA224 | 17  | MD5  | 14     | DJB      | 19423  |
| SHA384 | 18  | RS   | 11 609 | FNV      | 12 061 |
| SHA256 | 18  | BP   | 29 424 | CityHash | 20     |
| SHA512 | 21  | PJW  | 22 671 | FarmHash | 8      |
| SHA1   | 24  | ELF  | 22 671 |          |        |

综合本章所述, FarmHash 哈希函数可用于 IPv6 互联网与 IPv4 网络相互通信的两种场景中,具有处理时间短、冲突频率低、反向映射复杂度小等优点,验证了本文所述无状态机制的可行性。本文并未列举出所有的哈希算法,但其他可很好满足三项指标的哈希算法亦可应用于本文所述场景中。

## 4 结语

本文提出了 IPv4 网络与 IPv6 互联网双向无状态通信机制,是对已有无状态 IPv4/IPv6 翻译技术的有效补充,形成了统一的无状态 IPv4/IPv6 翻译框架。该机制的关键在于设计一个好的单向映射函数,因此本文提出了处理时间、冲突频率和反向查询复杂度三个定量的评价标准,并用实际数据对不同的哈希函数进行了分析和比较。实验表明, FarmHash 哈希函数能同时满足处理时间短、冲突频率低、反向映射复杂度小等要求,从而验证了该机制的可行性。与现有的有状态通信机制相比,该机制具有良好的可溯源性和网络运维性,可以很好地满足过渡过程中的需求,提供了一个平滑的 IPv4/IPv6 过渡路线。当 IPv6 用户大规模增加时,该 IPv4 网络应直接升级为 IPv6,使用 1.1 节所述无状态机制为 IPv4/IPv6 互联网提供服务。

(下转第 2123 页)

- control platform for large-scale production networks [C/OL]// Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2010 [2015-03-12]. [http://static.usenix.org/events/osdi10/tech/full\\_papers/Koponen.pdf](http://static.usenix.org/events/osdi10/tech/full_papers/Koponen.pdf).
- [6] CASADO M, FREEDMAN M J, PETTIT J, *et al.* Ethane: taking control of the enterprise [C]// SIGCOMM '07: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM, 2007: 1–12.
- [7] YEGANEH S H, GANJALI Y. Kandoo: a framework for efficient and scalable offloading of control applications [C]// HotSDN '12: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. New York: ACM, 2012: 19–24.
- [8] LIN P, BI J, HU H, *et al.* MSDN: a mechanism for scalable intra-domain control plane in SDN [J]. Journal of Chinese Computer Systems, 2013, 34(9): 1969–1974. (林萍萍, 毕军, 胡虹雨, 等. 一种面向 SDN 域内控制平面可扩展性的机制 [J]. 小型微型计算机系统, 2013, 34(9): 1969–1974.
- [9] BARI M F, ROY A R, CHOWDHURY S R, *et al.* Dynamic controller provisioning in software defined networks [C]// CNSM 2013: Proceedings of the 9th International Conference on Network and Service Management. Piscataway: IEEE, 2013: 18–25.
- [10] FU Y, JUN B, WU J, *et al.* A dormant multi-controller model for software defined networking [J]. China Communications, 2014, 11(3): 45–55.
- [11] YAO G, BI J, LI Y, *et al.* On the capacitated controller placement problem in software defined networks [J]. IEEE Communications Letters, 2014, 18(8): 1339–1342.
- [12] DIXIT A, HAO F, MUKHERJEE S, *et al.* Towards an elastic distributed SDN controller [C]// HotSDN '13: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2013: 7–12.
- [13] KRISHNA V. Auction theory [M]. New York: Academic Press, 2009: 30–39.
- [14] GAO L, XU Y, WANG X. MAP: multiauctioneer progressive auction for dynamic spectrum access [J]. IEEE Transactions on Mobile Computing, 2011, 10(8): 1144–1161.
- [15] KLOECK C, JAEKEL H, JONDRAL F. Auction sequence as a new resource allocation mechanism [C]// VTC-2005-Fall: Proceedings of the IEEE 62nd Vehicular Technology Conference. Piscataway: IEEE, 2005, 1: 240–244.
- [16] OpenFlow switch specification, version 1.4.0 [EB/OL]. [2015-02-11]. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>.
- [17] SCHRIJVER A. Theory of linear and integer programming [M]. Hoboken: John Wiley and Sons, 1998: 82–91.
- [18] LAUTZ B, HELLER B, MCKEOWN N. A network in a laptop: rapid prototyping for software-defined networks [C]// Hotnets-IX: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. New York: ACM, 2010: Article No. 19.

(上接第2117页)

#### 参考文献:

- [1] LI X, BAO C, CHEN M, *et al.* The China Education and Research Network (CERNET) IVI translation design and deployment for the IPv4/IPv6 coexistence and transition. IETF RFC 6219 [S/OL]. [2015-03-06]. [www.rfc-editor.org/rfc/rfc6219.txt](http://www.rfc-editor.org/rfc/rfc6219.txt).
- [2] BAO C, HUITEMA C, BAGNULO M, *et al.* IPv6 addressing of IPv4/IPv6 translators, IETF RFC 6052 [S/OL]. [2015-03-15]. [www.rfc-editor.org/rfc/rfc6052.txt](http://www.rfc-editor.org/rfc/rfc6052.txt).
- [3] LI X, BAO C, BAKER F. IP/ICMP translation algorithm, IETF RFC 6145. [2015-03-15]. [www.rfc-editor.org/rfc/rfc6145.txt](http://www.rfc-editor.org/rfc/rfc6145.txt).
- [4] LI X, MURAKAMI T. Specifications of MAP Translation (MAP-T) and MAP Encapsulation (MAP-E) [EB/OL]. [2015-03-05]. <http://www.ietf.org/proceedings/83/slides/slides-83-software-11.pdf>.
- [5] TSIRTISIS G, SRISURESH P. Network Address Translation-Protocol Translation (NAT-PT), IETF RFC 2766 [S/OL]. [2015-03-22]. [www.rfc-editor.org/rfc/rfc2766.txt](http://www.rfc-editor.org/rfc/rfc2766.txt).
- [6] AOUN C, DAVIES E. Reasons to move the Network Address Translator-Protocol Translator (NAT-PT) to historic status: IETF RFC 4966 [S/OL]. [2015-03-15]. [www.rfc-editor.org/rfc/rfc4966.txt](http://www.rfc-editor.org/rfc/rfc4966.txt).
- [7] BAGNULO M, MATTHEWS P, van BEIJNUM I. Stateful NAT64: network address and protocol translation from IPv6 clients to IPv4 servers, IETF RFC 6146 [S/OL]. [2015-03-22]. [www.rfc-editor.org/rfc/rfc6146.txt](http://www.rfc-editor.org/rfc/rfc6146.txt).
- [8] BAGNULO M, GARCIA-MARTINEZ A, van BEIJNUM I. The NAT64/DNS64 tool suite for IPv6 transition [J]. IEEE Communications Magazine, 2012, 50(7): 177–183.
- [9] IPv6 users by country [EB/OL]. [2015-03-02]. <http://labs.apnic.net/dists/v6dec.html>.
- [10] LEBER M. Global IPv6 deployment progress report [EB/OL]. [2015-03-02]. <http://bgp.he.net/ipv6-progress-report.cgi>.
- [11] BAO C, LI X, HAN G. A method for IPv6 clients accessing IPv4 servers: China, 103856580A [P]. 2014-06-11. (包丛笑, 李星, 韩国梁. 一种 IPv6 客户机访问 IPv4 服务器的方法: 中国, 103856580A [P]. 2014-06-11.)
- [12] IANA IPv4 address space registry [EB/OL]. [2015-03-02]. <http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml>.
- [13] RAJEEV S, GEETHA G. Cryptographic Hash functions: a review [J]. International Journal of Computer Science Issues, 2012, 9(2): 461–479.
- [14] PARTOW A. General purpose Hash function algorithms [EB/OL]. [2015-03-02]. <http://www.partow.net/programming/hashfunctions/>.
- [15] PIKE G, ALAKUIJALA J. Introducing CityHash [EB/OL]. [2015-03-02]. <http://googleo-pensource.blogspot.com/2011/04/introducing-CityHash.html>.
- [16] The FarmHash family of Hash functions [EB/OL]. [2015-03-02]. <https://code.google.com/p/FarmHash/>.
- [17] CZECH Z J, HAVAS G, MAJEWSKI B S. An optimal algorithm for generating minimal perfect Hash functions [J]. Information Processing Letters, 1992, 43(5): 257–264.
- [18] FAGIN R, NIEVERGELT J, PIPPENGER N, *et al.* Extendible hashing — a fast access method for dynamic files [J]. ACM Transactions on Database Systems (TODS), 1979, 4(3): 315–344.