



NDN 下 DHT 算法的设计与实现 *

雷 凯^{1,2}, 袁 梦^{1,3}

(1. 深圳市云计算关键技术与应用重点实验室 深圳 518055;

2. 北京大学大数据技术研究院 深圳 518055; 3. 北京大学信息工程学院 深圳 518055)

摘 要:设计了一种命名数据网络(NDN)下的分布式散列表(DHT)算法——ndnDHT,首次将 DHT 应用到 NDN 中;利用 NDN 的命名结构,提出了一种新的二进制命名方法,将 key 空间映射到 NDN 名字空间,以此实现 DHT 中的路由和转发等操作;设计了一种分配节点标识符的动态随机选择方法,使得 ndnDHT 算法的负载更加均衡;最后,基于 NS-3 和 ndnSIM 实现了 ndnDHT 算法,通过模拟实验验证了 ndnDHT 算法的可行性。

关键词:命名数据网络;分布式散列表;ndnDHT

doi: 10.3969/j.issn.1000-0801.2014.12.007

Design and Implementation of DHT Algorithm for NDN

Lei Kai^{1,2}, Yuan Meng^{1,3}

(1. Shenzhen Key Lab for Cloud Computing Technology & Applications, Shenzhen 518055, China;

2. Institute of Big Data Technologies, Peking University, Shenzhen 518055, China;

3. School of Electronics and Computer Engineering, Peking University, Shenzhen 518055, China)

Abstract: A novel distributed Hash table (DHT) algorithm under named data networking (NDN), which was called ndnDHT, was proposed. The algorithm brings the DHT to NDN for the first time. A new binary method using the structure of NDN was designed. By mapping the key space to the space of NDN, the binary method solves the operations such as routing and retransmission. Besides, a dynamic selection method of node identifier distribution was proposed to balance the load of ndnDHT algorithm. Finally, ndnDHT algorithm based on NS-3 and ndnSIM was implemented. The experimental result shows that the ndnDHT algorithm is effective.

Key words: named data networking, distributed Hash table, ndnDHT

1 引言

命名数据网络(named data networking, NDN)作为一种以内容为中心的新一代互联网网络架构,使用命名的内容而非终端的地址进行数据分组的路由。其完全崭新的通信模型给分布式应用的设计带来了许多重要的改变,然而对该架构下的分布式系统及其数据传输算法还未进行深入讨论。

分布式散列表(distributed Hash table, DHT)作为一种结构化数据检索的重要工具,得到了广泛应用,其代表协议有 Chord、CAN、Kademlia 等。基于这些 DHT 协议构建的 P2P 应用,因其良好的扩展性和健壮性,在分布式存储、即时通信等领域表现优异,快速推动了大规模分布式系统设计的发展。然而,近几年随着分布式系统规模的日益增大和激烈的行业竞争,许多全球化的互联网应用在保证高扩

* 国家自然科学基金资助项目(No. 61103027),国家发展和改革委员会基金资助项目(发改办高技[2013]1309号),深圳市科技计划基础研究资助项目(No. JSGG20140516162852628, No. JCYJ20130331144541058)

展性的同时,需要更迅捷的服务速度,如大型服务运营商的数据中心、高性能计算中心、云计算服务等。在分布式系统中,数据查询的效率和系统的扩展性一直是关注的焦点。为此,业界开发许多新的 DHT 算法及其相应的系统,如康奈尔大学设计的 Beehive 和亚马逊公司的 Dynamo 等,这些系统在基于传统 DHT 协议的 P2P 上进行了部分改进,在查询效率和系统扩展性之间寻找平衡。

因此,本文利用 NDN 架构上的特点设计相应的 DHT 算法。首次将 DHT 应用到 NDN 中,设计了一种 NDN 下的 DHT 算法——ndnDHT,为 NDN 的应用开发起到借鉴作用,同时也为基于 DHT 的算法设计提供新的视角;利用 NDN 的命名结构,提出了一种新的二进制命名方式,将 key 空间映射到 NDN 名字空间,使得 DHT 路由问题在 NDN 架构中得以有效解决,并设计了一种分配节点标识符的动态随机选择方式,使得 ndnDHT 算法负载更加均衡;基于 NS-3 和 ndnSIM 实现 ndnDHT 算法中节点设计,通过模拟实验验证了 ndnDHT 算法的可行性。

2 背景

2.1 DHT

在互联网发展的早期,基于客户—服务器架构的应用一直占据统治地位,然而随着系统规模的增大,中心化的体系架构已很难满足实际需求。由于 P2P(peer-to-peer)技术能够访问任意的分布式资源,最初被应用到文件交换系统中,后来逐渐成为分布式系统的设计原则和核心技术。基于 P2P 技术的分布式系统解决的核心问题是:在不需要中央服务器的情况下,能高效地查找、定位并管理数据;分布式的结构限制了系统规模的扩展性和高效性。DHT 便是在高效查询和大规模可扩展两者间寻找的解决方案。数据以<key,value>的形式存储在散列表中,DHT 在分布式系统中提供像散列表一样的服务,任何节点都可以进行存取、查询等操作。基于 DHT 的分布式系统具有如下 3 个特点。

- 去中心化:所有节点协同组成系统,而无中心连接。
- 错误容忍:当有节点增加、离开或发生错误时,系统仍可正常运行。
- 稳定性:当节点不断增加时,系统仍可正常运行。

DHT 有很多实现协议,具有代表性的协议如 Chord、Kademlia、Pastry、CAN、P-Grid、Apache Cassandra、Tapestry 等。在 IP 网络架构中,分布式系统中的 DHT 都是以 IP 地址作为数据的 key,并以此进行路由等操作。

2.2 NDN

虽然传统的 TCP/IP 通信网络占据着主导地位,但随着全球共享的数据丰富度越来越高,信息量级越来越大,加之用户更关注数据内容而非 IP 地址,人们逐渐开始尝试设计基于内容的网络架构。众所周知,IP 数据分组里携带了目的地址和源地址,但却无法很好地描述内容,因此研究人员试图用内容的名字代表地址作为唯一标识符,从而构建新的网络体系。NDN 便是一种以内容为中心的新一代互联网网络架构,使用内容来取代终端的地址,进行对数据分组的路由。它给现有的网络架构带来如下 3 点变化。

- 从开发角度,NDN 模型之上程序的开发只需要关注信息本身,而不用另外获取地址,由此可以省去很多中间件的开发和大量相关联的配置。
- 从安全角度,当前基于 IP 模型的安全措施基本上都是围绕会话通道构建,而目前的会话短暂又随机,对会话通道防护的手段并不满足当前以内容分发为主的互联网需求。在 NDN 中,所有数据分组都会从源头被签名和加密,因此 Web 上某人发布的网页或者视频文件会被唯一标识。
- 目前 IP 路由过程中为了防止 IP 分组的回路,只允许单出口转发,而 NDN 数据分组有唯一标识,因此可以通过路由记录放宽这一限制。NDN 也允许节点自由选择连接,增加了节点连通度。

一个完整的 NDN 系统包含 4 个部分:节点模型、命名机制、路由机制、转发机制,其通信模型以数据接收方作为驱动(如数据消费者),通过使用兴趣(interest)分组和数据(data)分组完成数据的传输,这些类型的分组都必须使用名字字段作为标识符,如图 1 所示。例如,消费者将需求数据的名字放入一个兴趣分组并将其置入网络中,兴趣分组利用此名字路由到数据生产者。一旦兴趣分组到达一个拥有请求数据的节点时,该节点便会将包含名称和内容的的数据分组连同生产者的签名,按照兴趣分组到来的路径返回给请求数据的消费者。

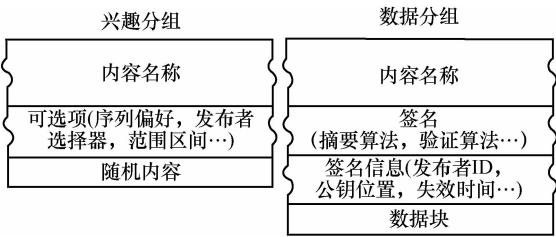


图 1 NDN 的兴趣分组和数据分组结构



NDN 是以层次化、结构化的字符串命名的,如“/parc.com/videos/WidgetA.mpg/_v<timestamp>/_s3”,用“/”划分名字,如图 2 所示,这种结构化形式利于上层应用表示模块之间的关系。NDN 的路由工作方式可以采用现在 IP 路由的设计,然而 NDN 的名字不限边界,按照名字产生的路由条目将会十分庞大,因此要设计相应的策略来控制路由条目的数量。当前的两种解决方案分别是使用预分配的名字、在大规模网络部署寻求小众世界的特征。后者因其具有高扩展性而被广泛使用。在 NDN 的转发机制里,最重要的是转发策略选择。转发策略作为关键功能使得 NDN 和 IP 相比具有相对优势。具体地说,NDN 为新来的兴趣分组动态选择多个接口,而不是只计算一条最佳路径。这种实时的决策机制使得节点能够最大可能地利用 NDN 丰富的连通性,通过路由以及转发机制防止数据分组被劫持攻击。

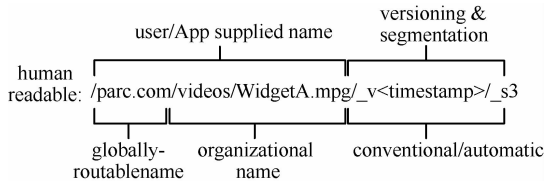


图 2 NDN 的名字结构

3 ndnDHT 算法

首先,从整体上介绍 ndnDHT 算法的设计概述和工作流程;然后,对 ndnDHT 算法中最关键的命名机制进行详述,并对标识符的分配给出了使负载更加均衡的动态随机选择方案;最后,给出节点上下线的相关操作。

3.1 设计概述及工作流程

ndnDHT 算法设计的关键同所有 DHT 算法一样,即对数据空间的划分及 key 的定位。在常见的 DHT 协议里,Chord 将空间看作环状结构,将 CAN 看作多维几何空间结构。ndnDHT 算法采取的划分方式类似 Chord,因为它简单且定位精准,更重要的是它会对后面的命名方式带来很大的便利。设定空间的划分方式之后,基于 IP 地址的 DHT 都需要自身在应用层维护不同节点的转发信息,以此构成多跳模型的中间站。而 NDN 的天然优势在于:一方面,它在路由层维护的是名字前缀信息,所以 ndnDHT 算法可将自身需要的路由信息与 NDN 路由信息统一结合,从而减少自身维护负担;另一方面,这种方式由于路由表所需维护信息相对较少,而且共享相同前缀的名字可以减少路由项,系统的可扩展性相应地提高。此外,ndnDHT 算法带来

的另一个优势是节点可以通过发布名字前缀来表示自身接收的数据范围,如/ndn/dht/music 代表了/ndn/dht/music/mylove.mp3、/ndn/dht/music/baby.mp3 等一系列数据内容,所以 ndnDHT 算法直接使用名字前缀标识自身,这与 Chord、CAN 相比,大大简化了设计。

图 3 简要介绍了 ndnDHT 算法的工作流程。首先,ndnDHT 算法中数据分组的名字通过散列操作得到唯一的 key,然后根据 NDN 的结构化命名特点转成二进制表达,并用树型结构描述。节点标识符生成的字符串与数据分组生成的二进制形式 key 进行匹配,一旦匹配上,则说明数据分组由本节点负责。ndnDHT 算法中,节点通过上线机制得到标识符,而标识符通过名字前缀发布到网络中。

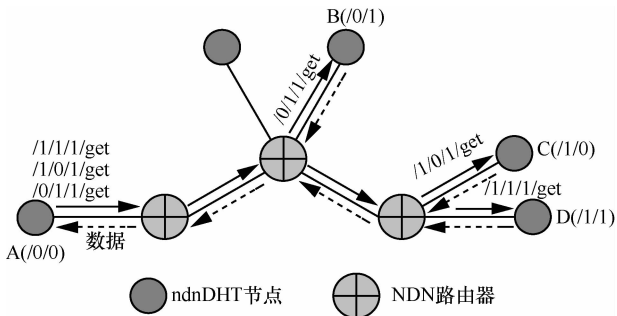


图 3 ndnDHT 算法请求数据

例如,图 3 中 A 节点对应标识符 0/0/0,则采用二进制字符串 0/0/0 表示 A 节点。同样 ndnDHT 算法中 B 节点对应标识符 0/0/1、C 节点对应标识符 1/1/0、D 节点对应标识符 1/1/1。ndnDHT 算法中关键字的表示与节点标识符类似,它是长度大于或等于节点标识符且大小固定的二进制字符串。图 3 中 A 节点先后发出 /1/1/1/get 等的 3 个请求兴趣分组,该名字表示 A 节点想获取关键字 /1/1/1、/1/0/1、/0/1/1 对应的数据分组。根据 NDN 的转发规则,B 节点将会接收兴趣分组 /0/1/1/get,然后 B 节点在本地检索相应的数据分组返回给 A 节点。与之类似,C 节点和 D 节点对收到的各自兴趣分组做出同样动作。可见,节点 A、B、C、D 根据自身发布的标识符各自独立、完整地负责了 3 位长度的二进制字符串表示的 key 空间。

当节点上线时,首先需要获得自身的标识符,该标识符在整个网络中是唯一的,由其他节点帮助或者随机产生。一旦获得标识符,随后便会通过配置本地路由表,以 NDN 前缀的方式发布到网络,从而由路由协议互相交换路由表,达到扩散节点信息的目的,最后其他节点可以访问到该节点的内容。当节点下线时,ndnDHT 算法寻找其他

负责离开该节点的标识符空间。如果节点是主动离开,那么可以转移本地储存的数据项。通过节点上下线时采用的方案,ndnDHT算法可以实现网络规模的动态变化,增加可靠性。

3.2 关键字空间模型

ndnDHT 算法同 Chord 一样,将 key 的范围限定在 $[0, 2^m - 1]$,其中 m 为 key 的位数,一般取 128 位。ndnDHT 算法生成 key 的方式是通过 SHA-1 函数作用于数据分组的名字产生的,为了充分利用 NDN 名字结构化的特点,本文不直接采取这种整数表示法构造名字,而是将 key 转化为 m 位二进制的表示形式,如 m 取 4, key 取 5 时, key 可以映射为 0/0/1/0/1/。节点标识符也是由 0、1 组成的 $n(n \leq m)$ 位字符串,即它可以是某 key 的子串。根据 NDN 名字最长前缀匹配的原理,一个节点 N 所管辖的数据范围就是所有与标识符匹配的 m 位长度的 key。可以看出,名字越短,管辖的数据范围越大;反之越小。例如, m 取 3,表示 key 取值范围是 $[0, 7]$,即从 0/0/0/0 到 1/1/1/1。图 4 中 A、B、C、D 表示节点标识符,而 key1、key2 等叶子节点表示关键字,其中叶子节点 B 和 C 既可表示 key,也可表示标识符。节点 B 只能

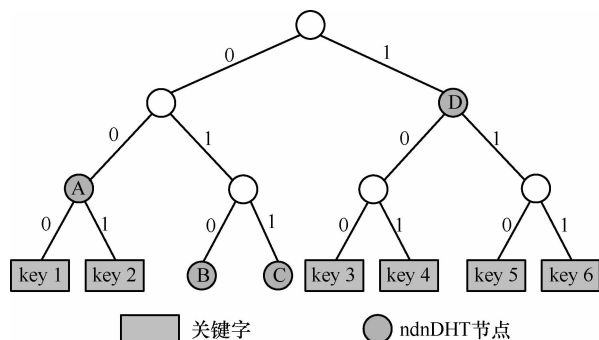


图 4 key 空间树型结构

管理 0/0/1/0 这个 key;标识符长度最短的 D 节点,负责其下所有子节点的 key。

3.3 节点标识符的分配

ndnDHT 算法将 key 空间看成树型结构,这与 Chord 的环形结构不同。当有节点上线时,获取节点标识符的过程即为添加子节点的过程。ndnDHT 算法中的 key 采取二进制字符串的表示形式,因此树型结构为二叉树。结合 NDN 前缀越短,负责的 key 范围越大的特点,ndnDHT 算法将获取节点标识符与负载均衡两个问题关联起来。为了使负载均衡尽可能均衡,便要寻找二叉树中与根节点之间路径最短的叶节点。此处介绍两个选择方式:单次随机选择和多次随机选择。

3.3.1 单次随机选择

假定 key 的长度为 m ,当节点加入系统时,节点可以随机产生 m 位 key,然后由该 key 构造兴趣分组,发送获取节点标识符请求。 $\text{key} = \text{SHA-1}(\text{string}(\text{MAC}) + \text{string}(\text{PID}) + \text{string}(\text{time}))$ 为产生 key 的一种基于 SHA-1 的方式。由于 NDN 不一定有 IP 地址,但一定存在硬件网络地址,此外,单个主机上允许多个 DHT 应用的存在,所以加入进程号 (PID) 这个变量。时间变量 time 保证每次随机出不同的 key。

由此,ndnDHT 算法就可以采用最简单的单次随机选择方式生成 key,选择叶子节点划分的方式。单次随机请求节点标识符的时序如图 5 所示,新节点 A 发送兴趣分组到节点 B,节点 B 收到请求标识符之后,重新构造标识符,然后节点 B 启动定时器等待节点 A 的成功返回分组。节点 A 获取标识符之后,在本地路由表添加路由表项,对网络声明了自身的存在。然后,需要将已经成功获取标识符的消

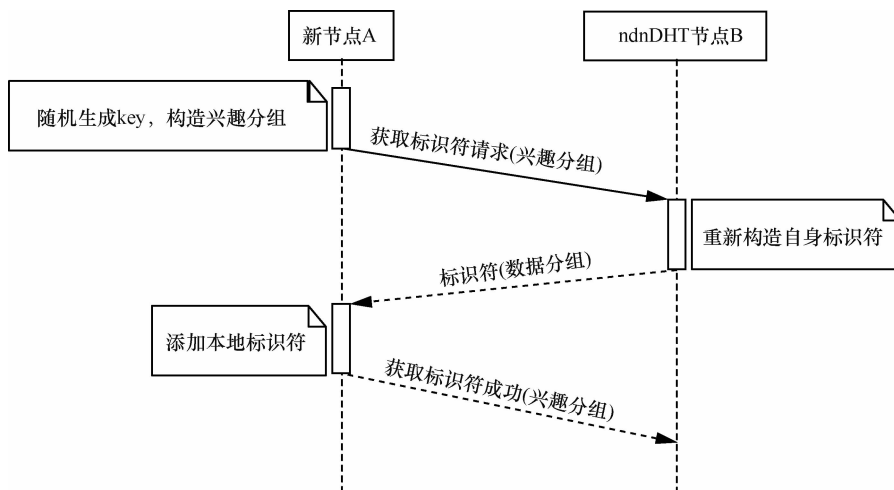


图 5 单次随机请求节点标识符时序



息告知刚才返回数据分组的节点 B, 节点 B 收到分组后同样需要更改自身标识符。若节点 B 在规定时间内未收到返回分组, 表明刚才的标识符分配失败, 则节点 B 依旧使用原来的标识符。

图 6 说明了 key 空间二叉树的更新过程, 其中 $m=4$, 假定产生的 key 为 0/0/0/1, 那么新节点 E 发送由该 key 构造的兴趣分组, 发送到标识符为 0/0/0 的 A 节点。A 节点收到分配标识符的请求兴趣分组, 返回自身标识符。节点 A 检测到节点 E 后更新自身标识符, 相当于在二叉树中添加左右子节点。此处 ndnDHT 算法规定: 新加入节点成为左子节点, 原先父节点退为右子节点。图中深灰色的节点 A 表示原来节点在树中的位置, 浅灰色的子节点 A 和 E 则表示标识符分配之后建立的新树。新节点 E 从节点 A 那里除了可以获取标识符之外, 还会获得节点 A 的后继节点列表。

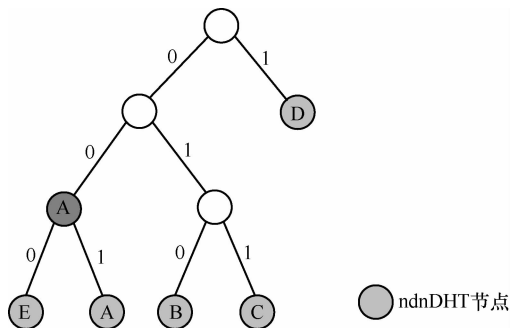


图 6 ndnDHT 算法中新节点获得节点标识符后的树

单次随机选择叶子节点的特点是随机性大, 假定网络规模为 n , 那么能选到最短前缀节点的概率为 $1/n$ 。为了增加得到最优节点的概率, 新加入节点可以采用多次产生 key 的方式, 多次发送兴趣分组请求标识符, 最后从返回的标识符中选取标识符长度最短的一个作为自身的 ID。如可以采用 3 次随机选择, 那么得到最优解的概率就增大为 $1-(n-1)^3/n^3$ 。毫无疑问, 随机选择的次数越多, 得到最优解的概率越大。

3.3.2 多次随机选择

单次随机选择方式中寻找各个节点的概率是均等的, 负载没有成为其考虑因素, 因此多次随机选择方式寻找负载高的节点, 有机会达到更好的效果。多次随机选择最优方法根据选择次数分为确定和非确定两类。确定次数的随机最优选择在大规模网络中可能导致分配极不均衡; 而非确定性方案则需要根据网络规模决定随机次数, 所以也称为动态随机选择。可以预见的是为保证一定的负载均衡性

能, 网络规模越大, 需要随机选择的次数越多。ndnDHT 算法对网络规模的预测可以通过其他网络节点前缀的长度进行判断。当新节点要加入网络时, 首先发送 3 个网络规模探测分组到其他节点, 其他节点收到该分组, 则返回自身前缀长度; 然后从 3 个返回的长度中选择最大值, 作为随机选择最优的次数; 最后, 按上述次数随机生成 key 请求自身标识符, 选择最优值。如图 7 所示, 新节点探测 3 次, 依次访问了节点 A、B、D, 得到最长标识符长度为 3。

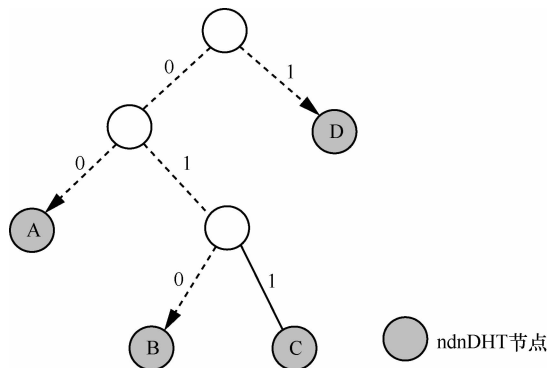


图 7 ndnDHT 算法探测方式预测网络规模

3.4 节点的加入与离开

3.4.1 节点加入

新节点的加入不仅涉及自身获取信息, 而且会引起系统其他节点的变化, 其详细流程分为: 通过 ndnDHT 算法获取节点标识符和前驱、后继节点列表; 新节点修改本地的网络路由表, 发布节点前缀; 新节点给前驱列表中的节点发送心跳; 新节点用自身后继列表的消息通知后继列表中的节点, 要求它们向自身发送心跳消息。新节点的前驱节点收到新节点的心跳后, 在后继列表队列中插入新节点, 维护其在线状态。而后继列表的节点收到新节点的后继列表, 更新本地的前驱列表, 然后开始给新节点发送心跳消息。此外, 新节点的标识符根据第 3.3 节中的方式进行分配。

3.4.2 节点离开

一个健壮的 DHT 系统必须考虑节点离开这种情况。ndnDHT 算法中节点失效离开的情况分为两类: 主动通知系统离开和出现故障离开。节点主动通知系统离开可以事先将自身负责的 key 转移到邻近节点, 并告知由它重新发布标识符扩大管辖 key 的空间范围, ndnDHT 算法中节点 key 空间范围的扩大通过发布新前缀来实现。节点主动离开系统可以避免数据丢失的情况发生, 但如果节点是出现

故障离开且没有备份数据,那么基于 ndnDHT 算法设计的数据复制功能将会提供帮助。具体地说,ndnDHT 算法采用了一种目前较为广泛使用的主从复制模型,如图 8 所示,其中主节点相当于 key 的主节点,每个 ndnDHT 算法中的节点既是主节点又是从节点。

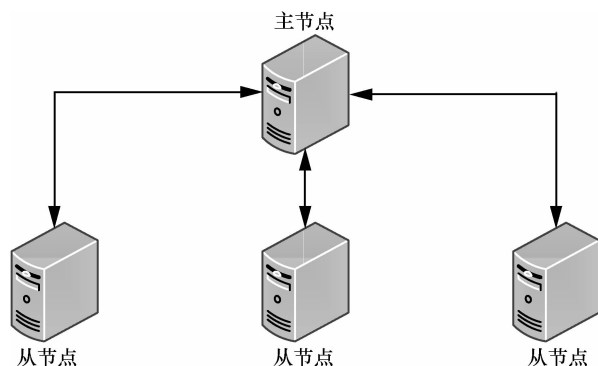


图 8 主从复制方式

该模型采用以关键字为单位的复制粒度,即每次客户端发起的 put 操作(将客户端原本作为数据消费者的身份转变为数据生产者)会发生复制动作。在选择备份节点时,ndnDHT 算法选择前驱或后继列表中的一个,进行复制操作。

4 实验与分析

实验部分主要考察 ndnDHT 算法的查询效率,以查询时延作为主要指标,put 操作作为附加考察对象。此外,比较前文中阐述的两种随机选择方法,以前缀长度的分布情况考察负载均衡效果。

在实验情景设计上,选取一个节点作为查询发起源,依次向网络发出 1 万个查询请求,最后统计每个请求分组从发出到收到反馈的平均用时。查询的字符串由用户随机生成,其长度固定为 10 byte,经过 SHA-1 的处理,均匀分布在 key 空间。返回的查询分组中数据长度固定为 20 byte。在实验中并没有事先给节点插入用户数据,采取方式如下:当有查询请求时,ndnDHT 算法将判断是否为自身负责的 key 空间范围,如果是,则构造查询请求反馈分组给查询者。这样做可以省去插入用户数据的步骤,而且对实验结果没有影响,put 操作的实验情景与之相同。此外,实验将单独对 ndnDHT 算法中的前缀分布进行考察,并对静态和动态两种情况下的前缀分布效果进行比较。

4.1 拓扑结构

拓朴生成器的出现解决了大规模网络图的产生问题,

但是对于较大规模的拓扑图,用户很难知道真实的拓扑情况。而 ndnDHT 算法作为 NDN 首个 DHT 算法,对网络测试的规模程度要求不是很高,但对整体的拓朴节点分布情况需要一定把握,因此选用点对点的网格模型,图 9 就是一种 4×4 的网格模型。其中,深色表示 ndnDHT 算法中的节点,浅灰色节点表示路由节点。模拟实验采用 NS-3 中的点对点模型连接生成 16×16 的网格模型,并将这 256 个节点都安装上 NDN 协议栈。然后,在网络上均匀部署 20、40、60、80、100、120 个应用节点,均匀部署指的是节点号均匀分布。图中 16 个节点均匀安装 3 个应用节点,它们分别在 1、6、11 号位置。为了减少拓扑结构对协议模拟效果的影响,应用同一网络拓朴均匀部署。

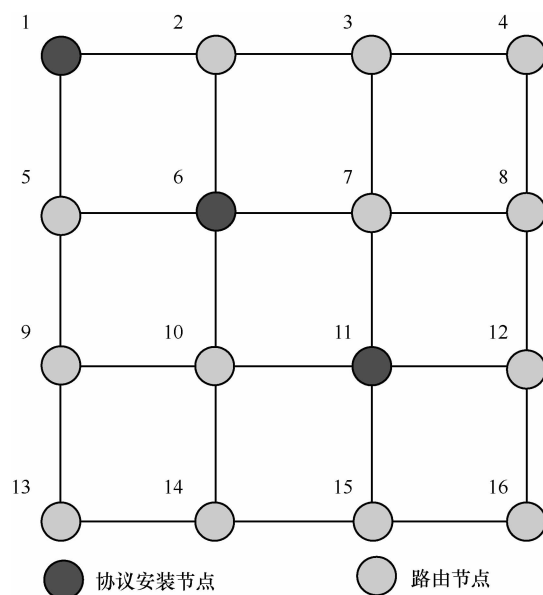


图 9 网络拓扑结构

4.2 实验结果

ndnDHT 算法从单一指定节点每隔 3 s 发出 key 随机生成的查询分组,将发出时刻和收到查询返回分组的时刻之差作为查询时延,然后统计 1 万个查询成功的请求,这里不包含失败的查询。实验中把查询源节点设定在网格中的一号节点,并且把标识符的分配方式选为单次随机选择。ndnDHT 算法的查询时延随节点数增多的变化情况如图 10 所示。从图 10 中可以看出,时延基本维持在 50 ms 左右,该实验将不同的数据应用节点均匀地分布在同一网络拓朴中,所以查询的平均跳数在 50 跳左右。ndnDHT 算法出现一定的浮动在于模拟实验过程 key 产生的随机性、节点部署数量上的不同。随机生成的 key 带来的影响是很

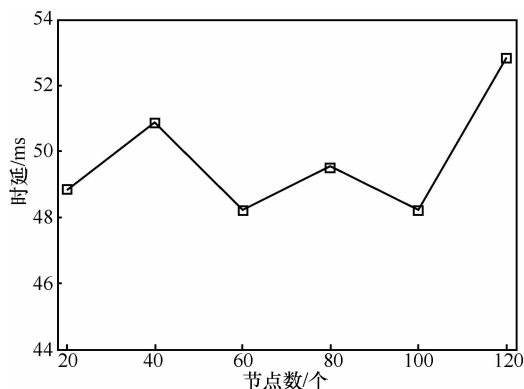


图 10 ndnDHT 算法中查询操作时延与节点数

微小的,统计意义上可以忽略不计。

为评测 ndnDHT 算法的负载均衡效果,在前述 16x16 网络拓扑部署 100 个网络节点,将部署之后的节点获取的标识符长度(不包括公共前缀、广播、单播前缀)加以统计,结果见表 1。可以看出节点上线采用单次随机选择标识符,导致负载均衡性很差,其长度分布范围为 5~9 个,幅度为 5 个单位长度,动态随机选择幅度为 3 个单位长度,最佳选择的方案(每次选择标识符长度最小加以均分)幅度为 2 个单位长度,这是由于其所构造的二叉树为完全二叉树,叶子节点之间高度差最大为 1。此外,随机选择越多,节点标识符获取的长度越均衡,因而负载越均衡。网络中部署 100 个应用节点,相当于构建 100 个节点的二叉树。显然,越均匀分布的二叉树,其叶子节点高度约集中在第 7 层。

表 1 标识符分配方式及其长度对应的节点数

标识符分配方式	标识符长度/个				
	5	6	7	8	9
单次随机丢失	5	28	38	27	2
动态随机选择		29	69	2	
最佳选择		28	72		

模拟实验评估了节点部署后静态情况(无节点失效)下的标识符分布情况,与节点加入和离开进行模拟对比。首先在网格中部署 100 个节点,然后先后随机让 50 个节点失效,在一个节点失效的同时,加入一个新节点,最后节点数依然维持在 100 个。图 11 对比了节点静态情况下和节点动态变化后标识符长度分布的标准差。从图 11 可以看出,单次随机选择、动态随机选择、最佳选择标识符方案的标准差依次减小,说明负载更加均衡。节点动态变化后,由于离开节点的 key 空间转移到前驱节点上,所以前驱节点的负载会随之大大增加,破坏系统整体的均衡性,导致标准差值的升高。

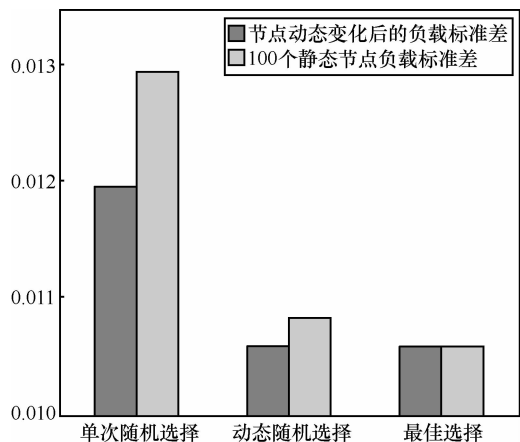


图 11 部署 100 个 ndnDHT 节点的系统上下线后的负载变化

5 结束语

本文提出了一个新一代网络架构 NDN 下的 DHT 算法——ndnDHT。ndnDHT 算法充分利用 NDN 中名字的结构化特点,设计了相应的关键字空间模型和节点标识符的分配方法,并给出节点上下线及数据的路由分发等操作。通过在 NS-3 和 ndnSIM 模拟平台上实验,证实了 ndnDHT 算法的可行性。

NDN 是当今研究的热点问题,其通信模型颠覆了以往 IP 网络点对点的方式。ndnDHT 算法是基于 NDN 重新设计 DHT 的一次积极尝试,它为 NDN 下的分布式系统数据管理提供了新的视角。NDN 下分布式系统的研究目前处于初级阶段,在负载均衡、安全隐私等方面有待更加深入的研究。

参考文献

- 1 Zhang L X, Estrin D, Burke J, *et al.* Named Data Networking (NDN) Project. Tech Rep, 2010
- 2 Ion S, Robert M, David K, *et al.* Chord: a scalable peer-to-peer lookup service for internet applications. Proceedings of ACM SIGCOMM, San Diego, USA, 2001: 149~160
- 3 Sylvia R, Paul F, Mark H, *et al.* A scalable content-addressable network. Proceedings of ACM SIGCOMM, San Diego, USA, 2001: 161~172
- 4 Petar M, David M. Kademlia: a peer-to-peer information system based on the XOR metric. Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Cambridge, USA, 2002: 53~65
- 5 Luiz A B, Urs H. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. San Rafael: Morgan & Claypool Publishers, 2009