命名数据网络下基于 K-medoids 的簇内 Hash 路由机制

鄢 欢,高德云,苏 伟
(北京交通大学电子信息工程学院,北京 100044)

摘 要: 命名数据网络(Named Data Networking,NDN)是以内容为中心的新型网络架构,其随处缓存策略存在缓存冗余过多、邻居缓存利用率低等问题,导致缓存空间的浪费及缓存效率的低下. 本文提出的融合沿路径非协作和路径外协作的缓存路由机制(K-Medoids Hash Routing,KMHR),使用 K-medoids 算法选取层次簇内的中心点,并针对不同流行度的内容分别采用 Hash 路由及最短路径路由,保证簇内高流行度内容的精确定位和唯一性,降低缓存冗余,提高缓存效率. 通过真实网络拓扑仿真得出,KMHR 机制具有最低的请求时间、最优的路由增益和较少的缓存内容数量.

关键词: 命名数据网络; 层次簇; K-medoids 算法; Hash 路由
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2017)10-2313-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.10.001

K-medoids Based Intra-Cluster Hash Routing for Named Data Networking

YAN Huan,GAO De-yun,SU Wei
(*College of Electronics and Information Engineering,Beijing Jiaotong University,Beijing* 100044,*China*)

Abstract: Named Data Networking (NDN) is a new content-centric architecture. Its original Leave Copy Everywhere (LCE) strategy has many shortcomings, for instance, massive cache redundancy and poor utilization of neighbors' cache, which waste cache space and lower cache efficiency. We proposed a routing scheme named K-Medoids Hash Routing (KMHR) combining on-path non-cooperation and off-path cooperation schemes. K-medoids algorithm is used to select some content routers as medoids in hierarchical clusters. And for different popularity of contents, we choose Hash routing or the shortest path routing separately. KMHR locates and insures the uniqueness of contents with high popularity in the cluster, which significantly reduces cache redundancy and improves cache efficiency. Based on the real world network topology, simulation results show that KMHR has the shortest request time, optimal routing gain and fewer cached contents.

Key words: named data networking; hierarchical cluster; K-medoids algorithm; Hash routing

1 引言

随着面向内容的应用极速增长,人们对于信息服务及数据内容的需求也不断增加. 根据思科的预测[1],到 2019 年,互联网视频相关的流量将占据所有消费者互联网流量的 80% 以上. 而现有互联网以主机为中心的网络架构及端到端的传输范式将难以满足未来信息服务发展的要求. 因此,信息中心网络(Information Centric Network,ICN)[2]将信息内容本身作为通信主体,不再关注内容的位置,是对现有互联网架构的变革. 其中,NDN[3]是 ICN 的一种典型架构,其用内容名称路由取代 IP 地址路由[4],并且路由器具有缓存功能(称为内容路由器),也可以响应用户的请求,从而减少用户的访问时间,降低内容源服务器的负载.

NDN 原始的缓存路由机制为随处缓存机制 LCE,即请求(Interest)包按照转发信息表(Forwarding Information Base,FIB)的条目寻找所需内容,并在待定兴趣表(Pending Interest Table,PIT)中记录 Interest 包的转发痕迹. 应答数据(Data)包沿着 Interest 包转发的反向路径返回,并依次缓存在内容路由器的内容缓存(Content

收稿日期:2016-06-15; 修回日期:2016-06-18;责任编辑:郭游
基金项目:国家 973 重点基础研究发展规划(No. 2013CB329100); 国家自然科学基金重点项目(No. 61232017)

Store,CS)中[3].但相对于数量庞大的内容而言,缓存空间极其有限.此外,每个内容路由器无区别的缓存内容将导致网络中存在大量的缓存冗余,不仅浪费稀缺的缓存空间,而且缓存的内容被频繁替换使得缓存效率低下.因此 LCE 机制虽易于部署但可行性较差.

为减少缓存冗余,提高缓存效率,国内外的研究人员提出了多种优化方案.这些方案根据路由及缓存的位置可以分为沿路径(on-path)与路径外(off-path),根据内容路由器之间是否协作可分为协作式及非协作式.上述的 LCE 机制即为沿路径非协作式机制.文献[5]提出了另一种沿路径非协作机制 Prob(p)即每个内容路由器采用固定的概率 p 缓存内容.

Probcache[6]通过动态估计路径的缓存容量得到内容路由器的缓存概率,使距离内容请求者近的内容路由器以更大的概率缓存内容,Probcache 属于沿路径协作式缓存路由机制.同类的机制还有 LCD[5],Betw[7],CATT[8].LCD 中只有缓存命中处的下游内容路由器缓存内容.Betw 要求内容仅缓存在传输路径上具有最大中介中心性(Betweenness Centrality)的内容路由器.CATT 机制利用势能路由转发请求,并将内容随机缓存在沿路径的某个内容路由器处,并相互通告缓存信息.

上述优化机制中不同路径的内容路由器独立地做出缓存决策,易产生重复缓存,因此文献[9]-[11]提出路径外协作式缓存路由方案来减少缓存冗余,提高邻近内容路由器的缓存利用.CoRC 机制[9]将内容名称 Hash 为固定长度的内容发布者标识(Publisher Identifier,PID),并根据 PID 进行分区,内容路由器通过查找 PID 转发请求,且只缓存属于本分区的内容.文献[10]针对视频流点播服务提出一种协作缓存策略,即将内容序号 Hash 到相应的内容路由器,并引入协作路由表及协作缓存表记录其他内容路由器的路由和缓存信息.SCAN 机制[11]的内容路由器通过扫描内容路由表找到邻近内容,同时采用布隆过滤器压缩需交互的内容路由表,缓解可扩展性问题.

以上文献设计的沿路径缓存路由机制无法有效利用路径外的缓存资源,而路径外缓存路由机制则缺乏对缓存通告开销及内容请求特征方面的考虑.因此本文采纳二者的优势,提出基于 K-medoids 的簇内 Hash 路由机制.该机制在层次簇网络架构下,通过执行 Hash 算法唯一地定位并缓存高流行度内容;对于低流行度的内容,则用最短路径路由获取,并以概率缓存在沿路径的非中心点中,从而降低缓存冗余,提高缓存命中.此外,通告消息的范围缩小到各簇,降低通告开销.

2 层次簇网络模型

本文提出的层次簇网络模型共有两层,分别为核心层和边缘层.其中,核心层路由器无缓存空间,仅提供快速转发和路由,且该路由器组成唯一的无簇首的核心簇(Core Cluster,CC).边缘层的路由器有缓存功能,尽可能地响应用户请求,且该内容路由器可分为多个边缘簇(Edge Cluster,EC),每个簇包含三类内容路由器:簇首,网关和成员,功能分别为:

簇首是其他成员及网关选举出来的簇控制器,功能是收集和维护内容流行度排序及 Hash 映射的更新.

网关是两个簇之间的边界内容路由器,功能是判断请求内容在本簇的流行度排序,维护 Hash 标志位.

成员是除了簇首和网关之外的内容路由器.成为接入内容路由器的成员同样计算请求内容的流行度排序,维护 Hash 标志位.其他成员则检测标志位,执行相应的路由机制.

内容提供者和内容请求者不是簇成员,因此二者的移动并不会影响簇结构,使得层次簇网络模型具有良好的移动性.

2.1 簇首选举

基于权重的边缘簇首选举算法是对文献[12]中方案的改进,如公式(1)所示.该算法包含三个影响因子:邻居节点度的倒数 d_v^{-1},传输时延 T_v 和权重跳数 H_v.为了确保计算是在相同维度下进行,我们将这三个影响因子进行归一化处理.最终具有最小权重的节点将被选举为簇首.

$$W_v=\alpha_1\frac{1/d_v}{avg(1/d)}+\alpha_2\frac{T_v}{avg(T)}+\alpha_3\frac{H_v}{avg(H)} \quad (1)$$

其中,$\alpha_1,\alpha_2,\alpha_3$ 是权重因子,且 $\alpha_1+\alpha_2+\alpha_3=1$.

在公式(1)中,d_v 是节点 v 的邻居节点数量.若 d_v^{-1} 越小,则说明节点 v 的邻居节点越多.邻居节点度倒数的平均值 $avg(d^{-1})=\frac{1}{V_0}\sum_{i=1}^{V_0}d_i^{-1}(i\in\Omega(V_0))$,$\Omega(V_0)$ 是所有节点的集合.

传输时延 T_v 定义为 $T_v=\frac{1}{M}\sum_{j=1}^{M}t_{vj}$,其中 t_{vj} 是从节点 v 到相邻节点 j 的传输时延($v\in\Omega(V_0)$,$j\in\Omega(M)$,$M<V_0$),$\Omega(M)$ 是节点 v 相邻节点的集合.显然,传输时延越小,散播消息的速度越快.同样,平均传输时延 $avg(T)=\frac{1}{V_0}\sum_{i=1}^{V_0}T_i(i\in\Omega(V_0))$.

权重跳数 H_v 描述了簇内的拓扑关系.为了减少收集和发布消息的开销,从簇首到其他簇内节点的跳数越小越好.因此权重跳数的计算公式为:

$$H_v=\beta_1\frac{1}{n_1}H_v^1+\beta_2\frac{1}{n_2}H_v^2+\beta_3\sum_{j=3}\frac{1}{n_j}H_v^j \quad (2)$$

其中 $n_1,n_2,\cdots,n_j$ 分别是到节点 v 的距离为 1 跳,2 跳和多跳(大于 2 跳)的节点数量.$H_v^1,H_v^2,\cdots,H_v^j$ 分别是与节

点 v 之间跳数为 1 跳,2 跳和多跳(大于 2 跳). 因此有 $H_v^1=1, H_v^2=2, H_v^j=j$. β_1,β_2,β_3 是跳数的权重因子,且 $\beta_1+\beta_2+\beta_3=1$. 为了使较小的跳数占的比重较大,权重因子之间的关系满足 $\beta_1>\beta_2>\beta_3$. 一种启发式的权重因子设置为 $\beta_1=0.5, \beta_2=0.33, \beta_3=0.17$. 平均权重跳数 $avg(H)=\frac{1}{V_0}\sum_{i=1}^{V_0} H_i(i\in\Omega(V_0))$.

根据上述公式,具有 $\min\{W_v\}$ 的节点被选举为簇首. 簇首的特点是:具有较多邻居节点,较小传输时延,较短权重跳数.

2.2　层次簇的建立

在簇首选举阶段后,簇首与其他节点之间可通过以下的信息交互过程建立层次簇结构. 另外,考虑到簇首的处理能力,我们设置了节点阈值 σ,即每个簇首可以管理的节点数量.

簇首发送 JOIN 消息到其他节点.

节点回复 PRE_JOIN 消息作为对加入该簇的响应.

簇首选出传输时延最短的 σ 个节点作为簇成员,并向此 σ 个节点发送 JOIN_ACK 消息作为确认其成为簇成员. 传输时延可以在通告 JOIN 消息和接收到 PRE_JOIN 消息的过程中进行测量.

收到 JOIN_ACK 消息的节点就是成员. 其中,与其他簇的成员之间有物理连接的成员被称为网关.

若节点没有加入任何簇,则该节点本身就成为簇首. 同时一些邻接簇的成员数量小于 σ 时,这些簇可以重新聚合.

3　基于 K-medoids 算法的内容路由器选取

本文的目标是设计一种融合了 on-path 和 off-path 的簇内路由机制 KMHR. 而如何选取用于 off-path 路由及缓存的内容路由器,以缩短请求时间,是本文首先需要解决的问题.

3.1　改进的 K-medoids 算法

聚类算法是将一组对象根据算法划分为多个类,达到同类对象的差别较小,不同类对象的差别较大的目的. 聚类算法因兼顾了公平性和效率,被广泛应用于网络选址问题,空间数据分析等领域. 其中 K-medoids 算法是通过最小化所有对象与已选定的最相近对象之间的差异度之和来确定 K 个中心点[13]. 这 K 个中心点是实际存在的对象. 下面本节给出基于 K-medoids 算法的内容路由器选取模型.

为了便于理解,给定一个边缘簇内的内容路由器集合 CR,设簇内内容路由器的数量为 V,有 $CR=\{cr_1, cr_2,\cdots,cr_V\}$, $|CR|=V$. $d(cr_i,cr_j)$ 是内容路由器 cr_i 与 cr_j 之间的差异度,当 $i=j$ 时,有 $d(cr_i,cr_j)=0$. 基于 K-medoids 算法的内容路由器选取最优化模型为:

$$\min\sum_{i=1}^{V}\sum_{j=1}^{V} d(cr_i,cr_j)x_{cr_i,cr_j} \tag{3}$$

$$\text{s.t.}\quad \sum_{i=1}^{V} x_{cr_i,cr_j}=1, j\in\{1,2,\cdots,V\}; i\neq j \tag{4}$$

$$x_{cr_i,cr_j}\leqslant y_{cr_i}, i,j\in\{1,2,\cdots,V\}; i\neq j \tag{5}$$

$$\sum_{i=1}^{V} y_{cr_i}=K, 1\leqslant K\leqslant V, K\in Z^+ \tag{6}$$

$$x_{cr_i,cr_j}, y_{cr_i}\in\{0,1\}, i,j\in\{1,2,\cdots,V\}; i\neq j \tag{7}$$

以上是 K-medoids 算法用在簇内内容路由器选取的数学模型. 其中约束条件(7)表示决策变量 x_{cr_i,cr_j} 及 y_{cr_i} 的取值区间为 0 或 1. 其中 $y_{cr_i}=1$ 表示内容路由器 cr_i 成为中心点,$x_{cr_i,cr_j}=1$ 代表内容路由器 cr_j 被分配到中心点 cr_i 所在的类(称为归属类). $i\neq j$ 的约束可减少变量的个数. 目标函数(3)要求每个内容路由器与中心点之间的差异度之和最小. 约束条件(4)表明,每个内容路由器 cr_j 只能被分配到一个归属类. 结合约束条件(7),对于给定的 cr_j,当 x_{cr_i,cr_j} 为 1 时,其他的 x_{cr_k,cr_j} 只能为 0. 约束条件(5)确保内容路由器 cr_j 不会被分配到非中心点所在的类. 当 $y_{cr_i}=1$ 时,结合约束条件(7),x_{cr_i,cr_j} 可为 0 或 1;而当 $y_{cr_i}=0$ 时,此时所有的 x_{cr_i,cr_j} 只能为 0. 约束条件(6)说明最终选出 K 个中心点用于 off-path 路由.

3.2　差异度的测量

在聚类算法中,差异度可用欧几里得距离测算,但在本方案中,欧几里得距离并不适合测量层次簇网络中内容路由器之间的距离. 鉴于 Floyd-Warshall 算法[14]可以计算网络中所有节点之间的最短路径,本节结合 Floyd-Warshall 算法给出了任意两个内容路由器之间的差异度,即最短路径的距离.

给定一个边缘簇内的内容路由器集合 CR,其中 $CR=\{cr_1,cr_2,\cdots,cr_V\}$, $|CR|=V$. 令 $dist(cr_i,cr_j,cr_m)_T$ 表示从 cr_i 到 cr_j 的最短路径的距离(这里用时延作为距离的度量),其中 $\{cr_1,cr_2,\cdots,cr_m\}$ 为这条最短路径通过的中间内容路由器集合,m 为中间内容路由器的数量.

对于 $cr_m=0$,当 $cr_i=cr_j$ 时,$dist(cr_i,cr_j,0)_T=0$;当 $cr_i\neq cr_j$ 且 cr_i,cr_j 直接相连时,$dist(cr_i,cr_j,0)_T=T(cr_i,cr_j)$,即 cr_i 到 cr_j 的时延;当 $cr_i\neq cr_j$ 且 cr_i,cr_j 不相连时,$dist(cr_i,cr_j,0)_T=+\infty$.

对于 $cr_m>0$,可分为两种情况:

(1)最短路径经过内容路由器 cr_m,则有

$$dist(cr_i,cr_j,cr_m)_T=dist(cr_i,cr_m,cr_{m-1})_T+dist(cr_m,cr_j,cr_{m-1})_T \tag{8}$$

(2)最短路径不经过内容路由器 cr_m,则有

$$dist(cr_i,cr_j,cr_m)_T=dist(cr_i,cr_j,cr_{m-1})_T \tag{9}$$

综合上述两种情况,cr_i 到 cr_j 最短路径距离递归公式为:

$$dist(cr_i,cr_j,cr_m)_T = \min\{dist(cr_i,cr_j,cr_{m-1})_T, dist(cr_i,cr_m,cr_{m-1})_T + dist(cr_m,cr_j,cr_{m-1})_T\} \tag{10}$$

对于任意的(cr_i,cr_j),用公式(10)依次计算当 $m=1,2,\cdots,V$ 时 $dist(cr_i,cr_j,cr_m)_T$ 的值,从而选出 $\min dist(cr_i,cr_j,cr_m)_T$ 作为(cr_i,cr_j)之间最短路径的距离 $dist(cr_i,cr_j)$.

3.3 内容路由器选取算法

基于 K-medoids 算法的内容路由器选取问题的最优化目标是最小化每个内容路由器与中心点之间最短路径的距离之和,最终选出 K 个中心点. 该 K-medoids 问题是一个 NP-难问题[15],已有文献给出一些近似算法,其中围绕中心点的划分(Partitioning Around Medoids,PAM)是最早提出的解决 K-medoids 问题的算法之一[16]. 该算法用非选定的对象替换已选定的对象,通过比较替换前后的总代价获取最优的中心点. 该算法不受噪音数据和孤立点的影响,且对于小数据集有较好的效果. 考虑到边缘簇内内容路由器的数量较少,因此结合本文的应用场景和 PAM 算法,本节设计了基于 K-medoids 内容路由器选取算法.

该算法包括两个阶段:初步选择阶段和替换确定阶段. 在初步选择阶段,依次选出 K 个内容路由器作为初始中心点,按 K 个中心点的选出顺序进行标号. 在替换确定阶段,交换中心点与非中心点来缩小差异度,同时更新标号.

同样地,边缘簇内的内容路由器集合为 CR,$|CR|=V$,设 CR^S 为中心点的集合,初始化 $CR^S=\varnothing$(即 $|CR^S|=0$),CR^U($CR^U=CR-CR^S$)为非中心点的集合. 基于 K-medoids 内容路由器选取算法最终从 CR 中选出 K 个中心点,即 $|CR^S|=K$,且 $K\leqslant V$. 因此,K 值决定了中心点集合的大小. 当 $K=V$ 时,说明 CR 中所有内容路由器都是中心点,有 $CR^S=CR$,此时根据内容路由器的名称进行标号. 当 $K=1$ 时,说明 CR^S 中只有一个中心点 cr_c 到其他内容路由器的距离之和最短. 因此 $CR^S=\{cr_c\}$,$|CR^S|=1$,cr_c 获得标号 $l(cr_c)$为 0.

当 $1<K<V$ 时,基于 K-medoids 内容路由器选取算法在初步选择阶段执行以下步骤:

步骤(1):在集合 CR 中,选择与其他内容路由器最短路径距离之和最小的内容路由器,将其作为第一个中心点 $cr_c(cr_c\in CR^S)$,此时 $|CR^S|=1$,cr_c 标号 $l(cr_c)=0$;

步骤(2):随机选择非中心点 cr_i 作为候选的中心点,其中 $cr_i\in CR^U$;

步骤(3):选择另一个非中心点 $cr_j(cr_j\in CR^U-\{cr_i\})$,令 cr_j 到$\widetilde{cr_c}$的距离是 cr_j 到 CR^S 中所有中心点距离中最短的,记为 $dist(cr_j,\widetilde{cr_c})$. 若 $CR^S=\{cr_c\}$,则有$\widetilde{cr_c}=cr_c$. 同时计算与 cr_i 的最短距离,记为 $dist(cr_j,cr_i)$;

步骤(4):比较 $dist(cr_j,\widetilde{cr_c})$ 与 $dist(cr_j,cr_i)$;

a)若 $dist(cr_j,\widetilde{cr_c})>dist(cr_j,cr_i)$,说明 cr_j 为 cr_i 成为中心点起正向作用,$g(cr_j,cr_i)=dist(cr_j,\widetilde{cr_c})-dist(cr_j,cr_i)$为当前的增益;

b)若 $dist(cr_j,\widetilde{cr_c})\leqslant dist(cr_j,cr_i)$,则说明 cr_j 起反向作用,此时的增益 $g(cr_j,cr_i)=0$.

因此综合以上两种情况,有:

$$g(cr_j,cr_i)=\max\{dist(cr_j,\widetilde{cr_c})-dist(cr_j,cr_i),0\} \tag{11}$$

步骤(5):计算 cr_i 为候选中心点的累积增益:

$$G(cr_i)=\sum_{cr_j\in CR^U} g(cr_j,cr_i) \tag{12}$$

步骤(6):将具有最大累积增益(即 $\max\limits_{cr_i\in CR^U} G(cr_i)$)的 cr_i 加入中心点集合. 更新 $CR^S=CR^S\cup\{cr_i\}$,$CR^U=CR^U-\{cr_i\}$. 将 $|CR^S|$加 1,cr_i 标号 $l(cr_i)=|CR^S|-1$.

步骤(7):当 $|CR^S|<K$ 时,循环步骤(2)到步骤(6),当 $|CR^S|=K$ 时,算法结束.

在替换确认阶段,我们考虑内容路由器替换对(cr_i,cr_h),其中 $cr_i\in CR^S$,$cr_h\in CR^U$,cr_i 的标号为 $l(cr_i)$. 替换 cr_i 和 cr_h 指替换后 $cr_i\in CR^U$,$cr_h\in CR^S$. 替换的代价 $C(cr_i,cr_h)$指,替换 cr_i 和 cr_h 后,每个非中心点与最近的中心点的距离差之和. 替换确认阶段的执行步骤如下:

步骤(1):选择非中心点 $cr_j(cr_j\in CR^U-\{cr_h\})$;

步骤(2):针对 cr_j,计算替换 cr_i 和 cr_h 的代价 $c(cr_j,cr_i,cr_h)$. 分为以下三种情况;

a)当 $dist(cr_j,cr_i)>dist(cr_j,\widetilde{cr_c})$ 且 $dist(cr_j,cr_h)>dist(cr_j,\widetilde{cr_c})$时,此时 $c(cr_j,cr_i,cr_h)=0$;

b)当 $dist(cr_j,cr_i)>dist(cr_j,\widetilde{cr_c})$ 且 $dist(cr_j,cr_h)<dist(cr_j,\widetilde{cr_c})$ 时,有 $c(cr_j,cr_i,cr_h)=dist(cr_j,cr_h)-dist(cr_j,\widetilde{cr_c})$;

c)当 $dist(cr_j,cr_i)=dist(cr_j,\widetilde{cr_c})$时,此时又分两种情况:

Ⅰ. 若 $dist(cr_j,cr_h)<dist(cr_j,\widetilde{cr_{c2}})$,其中$\widetilde{cr_{c2}}(\widetilde{cr_{c2}}\in CR^S-\{\widetilde{cr_c}\})$也是中心点,$dist(cr_j,\widetilde{cr_{c2}})$是 cr_j 到 CR^S 中所有中心点的次短距离. 替换的代价 $c(cr_j,cr_i,cr_h)=dist(cr_j,cr_h)-dist(cr_j,\widetilde{cr_c})$;

Ⅱ. 若 $dist(cr_j,cr_h)\geqslant dist(cr_j,\widetilde{cr_{c2}})$,此时代价 $c(cr_j,cr_i,cr_h)=dist(cr_j,\widetilde{cr_{c2}})-dist(cr_j,\widetilde{cr_c})$.

结合上述两种情况,有:

$$c(cr_j,cr_i,cr_h)=\min\{dist(cr_j,cr_h),dist(cr_j,\widetilde{cr_{c2}})\}-dist(cr_j,\widetilde{cr_c}) \tag{13}$$

步骤(3):计算替换 cr_i 和 cr_h 时的总代价为:

$$C(cr_i,cr_h)=\sum_{cr_j\in CR^U}c(cr_j,cr_i,cr_h) \tag{14}$$

步骤(4):对于任意内容路由器替换对(cr_i,cr_h),计算最小代价 $\min\limits_{cr_i\in CR^S,cr_h\in CR^U}C(cr_i,cr_h)$;

步骤(5):判断最小代价.

a)若 $\min\limits_{cr_i\in CR^S,cr_h\in CR^U}C(cr_i,cr_h)<0$,则用 cr_h 替换 cr_i,将 cr_h 的标号设为 $l(cr_h)=l(cr_i)$,删除 cr_i 的标号. 循环步骤(1)到步骤(5);

b)若 $\min\limits_{cr_i\in CR^S,cr_h\in CR^U}C(cr_i,cr_h)\geqslant 0$,则说明替换并不能使代价减少,因此不执行替换. 循环步骤(1)到步骤(5).

步骤(6):当遍历所有替换对后不再发生替换,即所有最小代价均大于 0 时,算法结束.

基于 K-medoids 内容路由器选取算法在替换确认阶段考虑了所有可能的替换对,替换对选取的先后顺序并不会影响算法的结果,因此该策略具有良好的健壮性.

3.4　中心点选取过程的示例

下面以图 1 所示的局部拓扑图为例来描述中心点的选取过程,最终选出 2 个中心点,具体过程如下:

(1)初步选择阶段:选择与其他内容路由器最短路径距离之和最小的内容路由器 cr_c 作为第一个中心点,有 $\widetilde{cr_c}=cr_c$. 由图 1 可知,cr_j 到 $\widetilde{cr_c}$ 的距离 $dist(cr_j,\widetilde{cr_c})$ 为 7ms,将 cr_i 作为候选中心点,与 cr_j 的距离 $dist(cr_j,cr_i)$ 为 3ms,因此有增益 $g(cr_j,cr_i)$ 为 4ms. 同理,对于另一个非中心点 cr_h,有 $g(cr_h,cr_i)$ 为 4ms. 因此累积增益 $G(cr_i)$ 为 8ms. 而将 cr_j 和 cr_h 作为候选中心点时,累计增益 $G(cr_j)$ 及 $G(cr_h)$ 分别为 2ms 和 6ms. 因此 cr_i 加入中心点集合,并赋予标号 1.

(2)替换确认阶段:替换 cr_i 和 cr_h 后,选取非中心点 cr_j,有 $dist(cr_j,cr_i)$ 为 3ms. 此时 $\widetilde{cr_c}=cr_i$ 且 $\widetilde{cr_{c2}}=cr_c$,$dist(cr_j,\widetilde{cr_c})$ 为 3ms,因此 $dist(cr_j,cr_i)=dist(cr_j,\widetilde{cr_c})$. 而 $dist(cr_j,\widetilde{cr_{c2}})$ 为 7ms,$dist(cr_j,cr_h)$ 为 4ms,替换代价 $c(cr_j,cr_i,cr_h)$ 为 1. 由于没有其他非中心点,因此最小代价 $\min\limits_{cr_i\in CR^S,cr_h\in CR^U}C(cr_i,cr_h)$ 为 1,最终不执行替换. 同理,也不执行替换 cr_i 和 cr_j

3.5　时间复杂度分析

给定所有内容路由器集合为 CR_o,$|CR_o|=V_o$,其中核心层路由器的数量为 V_C,边缘层分为 M 个簇,假设各簇中心点数量为 $K(K\leqslant V)$. 为了方便计算,令每个边缘簇的内容路由器数量相同,即 $|CR|=V=\lfloor\frac{V_o-V_C}{M}\rfloor$.

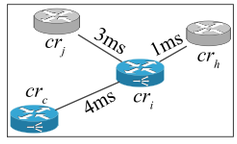


图1　中心点选取过程

文献[14]中最短路径算法 Floyd-Warshall 时间复杂度为:

$$O(M\lfloor\frac{V_o-V_C}{M}\rfloor^3) \tag{15}$$

针对单个边缘簇,初步选择阶段的时间复杂度为 $O(K)$. 替换确认阶段需要循环 $K(V-K)$ 个内容路由器对,且替换过程中还涉及$(V-K)$个非中心点协助计算,因此替换确认阶段的时间复杂度为 $O(K(V-K)^2)$[13]. 考虑到初步选择阶段的时间复杂度可以忽略,因此总时间复杂度为:

$$O(MK(\lfloor\frac{V_o-V_C}{M}\rfloor-K)^2) \tag{16}$$

结合上述公式(15)和(16),基于 K-medoids 内容路由器选取算法的时间复杂度为:

$$O(M\lfloor\frac{V_o-V_C}{M}\rfloor^3)+O(MK(\lfloor\frac{V_o-V_C}{M}\rfloor-K)^2)=O(M\lfloor\frac{V_o-V_C}{M}\rfloor^3) \tag{17}$$

若不考虑层次簇网络结构,整个网络的时间复杂度为:

$$O(V_o^3)+O(MK(V_o-MK)^2)=O(V_o^3) \tag{18}$$

综上,层次簇结构能够降低算法的运算时间.

4　基于内容流行度的簇内 Hash 路由机制

根据文献[17],应用 Hash 算法可实现唯一地定位内容,即 Interest 请求依据 Hash 算法被重定向到某些特定的内容路由器来获取内容. 虽然这种隐式协作的缓存路由方式降低了内容的冗余,但若要重定向每个 Interest 请求,也会在一定程度上增加传输开销,这是因为与本地接入路由器相比,特定内容路由器离内容请求者的距离更远.

为了兼顾缓存的唯一性和传输开销,本节提出基于内容流行度的簇内 Hash 路由机制,该机制与基于 K-medoids 的内容路由器选取算法相结合,对于流行度高的内容,采用 Hash 算法进行精确定位,且内容唯一缓存在 K 个中心点. 同时,为减少 Hash 路由的开销,对于低流行度内容的请求只按照最短路径路由,且内容按照一定概率缓存在沿路径的内容路由器.

4.1　簇内 Hash 路由机制的设计

内容流行度是由与内容请求者直接相连的接入内

容路由器根据内容请求的频率测量的. 每个簇的接入内容路由器将统计的结果发送给簇首,簇首按内容请求频率从高到底的顺序将内容进行排序,并将排序结果分发给所有内容路由器. 令内容的数量为 R,某内容 j 流行度序号为 $P_j \in \{0,1,\cdots,R-1\}$. 定义流行度高的内容序号 P_h 和流行度低的内容序号 P_l 分别为:

$$P_h \in \begin{cases} \{0,1,\cdots,K \cdot CS_{size}-1\} & K \cdot CS_{size} \leqslant R \\ \{0,1,\cdots,R-1\} & \text{else} \end{cases} \tag{19}$$

$$P_l \in \begin{cases} \{K \cdot CS_{size}, K \cdot CS_{size}+1,\cdots,R-1\} & K \cdot CS_{size} \leqslant R \\ \varnothing & \text{else} \end{cases} \tag{20}$$

其中,CS_{size}为内容路由器 CS 的缓存空间大小.

根据上节的分析,每个中心点都拥有唯一的标号 $l(cr_i)$,该标号的集合记为 $L(CR^s)$,有 $L(CR^s)=\{0,1,\cdots,K-1\}$. 簇内的内容路由器具有相同的 Hash 映射函数,参数分别为内容流行度排序后的序号及 K 值,将 Hash 结果与中心点的标号一一映射,如公式(21)所示.

$$l(cr_i) = \text{Hash}(P_j, K) \tag{21}$$

结合公式(19),(20),(21)有:当中心点的缓存空间之和不大于内容数量时,以 $K \cdot CS_{size}$作为内容流行度高低的边界值,序号不大于 $K \cdot CS_{size}$的 Interest 包/Data 包应按照 Hash 路由/缓存到相应的中心点;而其余的请求按照最短路径路由转发到内容提供者,同时,内容按概率缓存到沿路的非中心点. 当中心点的缓存空间之和大于内容数量时,流行度低的内容序列为空集,此时 Interest 包/Data 包都 Hash 路由/缓存到中心点. 但在真实网络中,内容的数量远大于内容路由器的缓存空间,因此在后续的分析中,我们只考虑第一种情况.

内容路由器新增了 Hash 路由表(Hash Routing Table,HRT)用于 Hash 查找,该表包含中心点标号(简称标号)及下一跳接口(即出接口). 为了区分 Hash 路由和最短路径路由,以及避免由 Hash 映射带来的重复 Interest 包问题,我们在 Interest 包格式中增加 Hash 标志位. 当该标志位置 1 时,说明请求内容的流行度高,执行 Hash 路由;否则,执行最短路径路由. 各内容路由器执行的查询顺序依次为:Hash 标志位,CS,PIT,FIB/HRT.

4.2 基于内容流行度的簇内 Hash 路由机制的应用

下面以图 2 所示的拓扑图为例介绍基于内容流行度的簇内 Hash 路由机制. 图 2 以两条虚线区分两个边缘簇 EC_1 和 EC_2,中间为核心簇 CC. 设 K 值为 2,EC_1 包含 5 个内容路由器,即有两个中心点 cr_4 和 cr_5,标号分别为 1 和 0,cr_1,cr_2,cr_3 为非中心点. 在 EC_2 中,cr_6,cr_7 为中心点,标号分别为 1 和 0,cr_8,cr_9,cr_{10}为非中心点.

当内容的流行度高且在本簇内无缓存(内容没有缓存在簇 EC_1 的 cr_4)而邻簇有缓存的内容路由过程如下:

(1)内容请求者发送 Interest 包请求的内容名称为“bjtu/a”,初始化 Hash 标志位为 0.

(2)当 cr_2 收到 Interest 包后,判断内容的流行度排序为高,将 Hash 标志位置 1,进行 Hash 映射得到内容“bjtu/a”对应的中心点标号为 1,而 cr_2 无标号,与查询的标号不符,因此无需搜索 CS. 再检查 PIT 表,若 PIT 表中没有匹配的条目,将该内容名称“bjtu/a”、收到该 Interest 的接口(即入接口)0 加入 PIT 表. 查找 HRT 表找到标号 1 对应转出接口(即出接口)1,将 Interest 包从接口 1 转发,并将出接口 1 添加到 PIT 表相应条目的出接口列表.

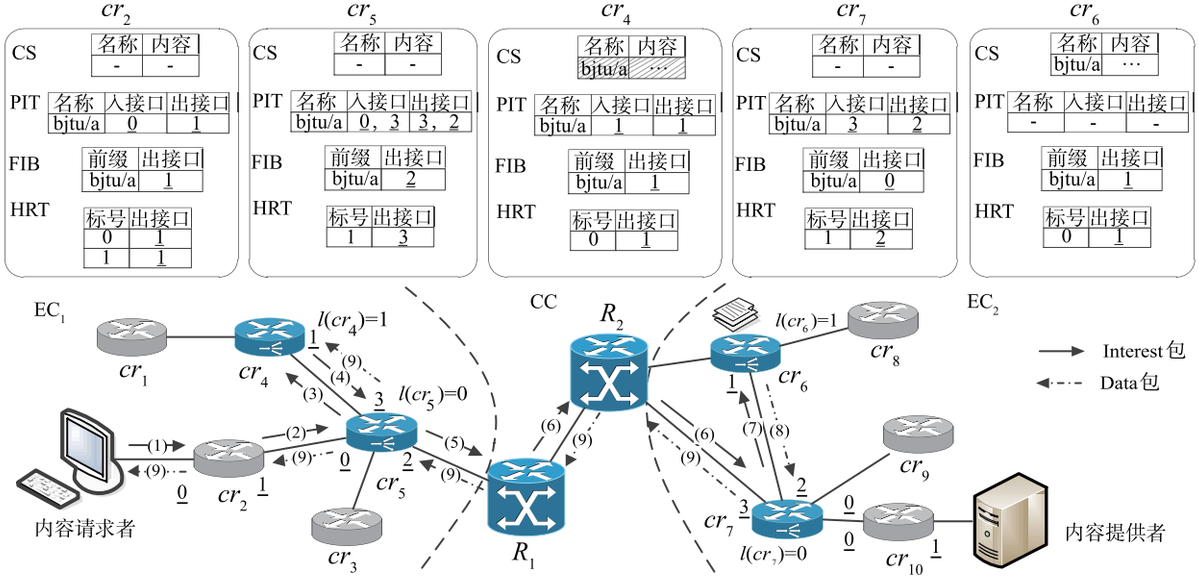


图2 簇内Hash路由流程图

(3)cr_5 收到 Interest 包后,检测 Hash 标志位为高,Hash 映射的中心点标号仍为 1,与 cr_5 的标号不符,同样无需搜索 CS. 查询并将信息添加到 PIT 表,按 HRT 表转发 Interest 包.

(4)接收到 Interest 包后,cr_4 检测到 Hash 标志位为 1,Hash 得出的中心点标号与本机的标号一致. 但查询 CS 后,并没有找到所请求的内容. 此时需将 Hash 标志位置 0,说明通过 Hash 路由没有获取到所需内容,转而执行最短路径路由. 搜索并添加 PIT 表,最后查询 FIB 表,而非 HRT 表,将 Interest 包按 FIB 表的出接口 1 转发.

(5)cr_5 的接口 3 收到请求后,发现在步骤(3)中 cr_5 已将相同的 Interest 包从接口 3 转发出去,因此若没有 Hash 标志位的区别,cr_5 会认为网络中出现回环,同时认定步骤(3)和步骤(5)中收到的 Interest 包是重复的,从而丢弃该包. 显然在 KMHR 机制中这种丢弃是不合理的,因此 Hash 标志位的设计避免了不必要的 Interest 包重复问题的出现. 当 cr_5 检测到 Hash 标志位为 0 时,首先检查 CS,若无缓存,则搜索 PIT 表. 当 PIT 表中存在匹配的内容条目时,继续检查入接口. 发现当前的接口 3 与已有入接口 0 不同,将接口 3 添加到该条目. 最后按照 FIB 表条目从接口 2 转发该 Interest 包.

(6)由于核心簇 CC 的路由器无缓存功能,因此 R_1 和 R_2 在接收到 Interest 包后无需搜索 CS,直接在 PIT 表中查询并添加相应条目,根据 FIB 表将 Interest 包转发到簇 EC_2 中 cr_7(KMHR 机制不存在缓存通告,所以 R_2 并不知道与其直连的 cr_6 可能缓存该内容的备份).

(7)簇 EC_2 的网关节点 cr_7 收到 Interest 包后,首先判断该内容在簇 EC_2 的流行度范围. 若属于高流行度,则重新将 Hash 标志位重新置 1. Hash 映射后得到的标号与本机不符,则无需查询 CS,依次搜索 PIT 表及 HRT 表并执行相应操作.

(8)cr_6 收到 Interest 包后,检测到 Hash 标志位置 1,Hash 返回的标号与本机的相同,则搜索 CS. 若查找到所需的内容“bjtu/a”,发送 Data 包作为响应,执行步骤(9). 若 cr_6 中没有备份,则执行类似步骤(4)的流程,这部分不再重复.

(9)Data 包按照内容路由器的 PIT 信息返回内容请求者. 由于 cr_5 的 PIT 条目中存在两个入接口 0 和 3,因此 cr_5 在向 cr_2 转发 Data 包的同时,复制一份发往 cr_4,从而保证请求者收到所需内容,中心点 cr_4 也缓存了内容的备份(如图 2 中用斜杠标出 cr_4 缓存的内容). 此次请求过程结束.

对于流行度低的内容,Interest 包的 Hash 标志位始终为 0,内容路由器按最短路径路由将请求转发至内容提供者,Data 包在沿路径的非中心点处按一定概率缓存,直到返回请求者.

5　仿真验证及分析

我们在 ndnSIM[18] 平台上对 KMHR 机制进行仿真实现. 仿真拓扑参照 AS3967 真实网络拓扑,链路带宽、路由度量及链路时延等参数都基于 Rocketfuel 的追踪结果[19]. 该拓扑由 192 个路由器组成. 依据层次簇架构的设计,39 个骨干路由器位于核心簇,无缓存功能;58 个网关路由器及 95 个边缘路由器共形成了 8 个边缘簇,每个簇的节点阈值 σ 为 25.

5.1　参数设置

内容请求者与内容提供者的数量各为 32,内容的总量为 80,000. 每次仿真时长为 500 秒,每秒产生的内容请求为 100,内容请求的速率服从泊松分布[18]. 相对缓存空间是指单个内容路由器的缓存空间占各内容请求者请求内容数量的百分比,相对缓存空间的变化范围是 0.5%,1%,2%. 本节还探讨了中心点数量 K 为 6,8,10 时对性能的影响. 在仿真的初始化阶段,各簇选出 K 个中心点,接入内容路由器统计内容的请求频率,簇首收集该频率,对流行度排序并通告簇内节点.

本文引入下面三种度量值来评估性能.

平均请求时间:即从发送 Interest 包到收到请求的 Data 包的平均传输时间. 该值可以表明缓存的优势.

路由增益:结合传输跳数及缓存命中,本节提出了路由增益的计算公式,如公式(22)所示. 其中 $\overline{hop_{req\to router}}$ 为从内容请求者到缓存命中的内容路由器之间的平均跳数,I_{hit} 为所有内容路由器缓存命中的数量,$\overline{hop_{req\to pro}}$ 为从内容请求者到内容提供者的平均跳数,I_{all} 表明总共的请求数量. 该路由增益可用于衡量路由机制的效率.

$$G(cache)=1-\frac{\overline{hop_{req\to router}}\cdot(I_{all}-I_{hit})}{\overline{hop_{req\to pro}}\cdot I_{all}} \tag{22}$$

缓存内容数量:即所有内容路由器缓存内容的总和,体现了缓存冗余提升情况.

为了验证 KMHR 机制的性能,我们还实现了四种缓存路由机制,包括两种非协作式(LCE[3] 及 Prob(p)[5])及两种协作式(LCD[5] 及 Betw[7]. 与 KMHR 机制中低流行度内容的缓存概率相同,Prob(p)中内容路由器的缓存概率也为 0.1.

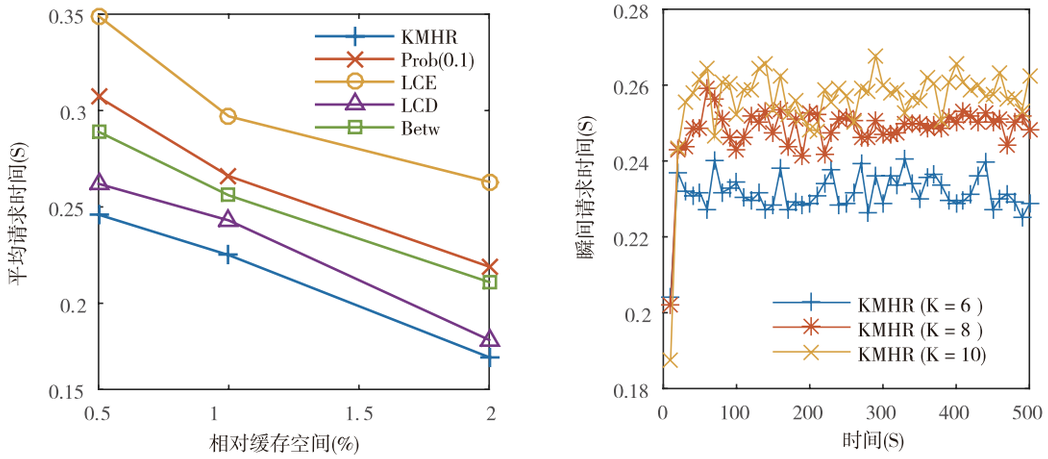
5.2　结果分析

下面比较这五种机制在平均请求时间、路由增益、缓存内容数量三个方面的优化情况,并分析相对缓存空间及中心点 K 的变化对性能的影响. 仿真中参数的设置基准是:相对缓存空间为 1%,Zipf 分布的参数 α 为 0.85,中心点 K 为 6.

5.2.1　平均请求时间优化

从图 3(a)可以看出,五种机制的平均请求时间都

随着相对缓存空间增加而降低. 相对缓存空间的增加使得内容路由器可存储的内容增加,因此更多的请求将在中心点或沿路的内容路由器处获取到所需内容. 由于 KMHR 机制将流行度高的内容唯一地分布在中心点中,从而将大部分的请求限制在簇内,在很大程度上降低了请求时间. 其中 KMHR 机制的平均请求时间与 LCE 相比降低了 29.51%(如图 3(*a*)中相对缓存空间为 0.5%时). 图 3(*b*)表明瞬时请求时间随 *K* 值的增加而增加. 当 *K* 为 6 时,瞬时请求时间最小,与 *K* 为 8、10 的情况相比分别降低了 9.27% 和 11.76%. 当 *K* 值增加时,新增的中心点距离其他内容路由器的距离更远,因而将 Interest 包定向到新增的中心点会延长请求时间.



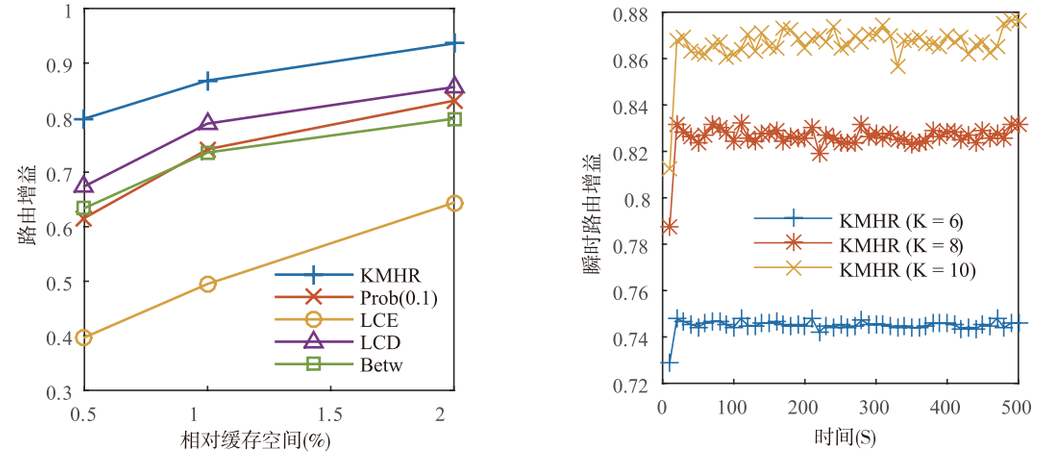
(a) 相对缓存空间的影响 (b) 中心点*K*的影响

图3 请求时间的优化

5.2.2 路由增益优化

图 4 中路由增益由公式(22)计算得出. 图 4(*a*)阐明了 KMHR 机制在路由增益方面的优势. 该机制采用 Hash 路由增加高流行度内容的可达性,提高了缓存命中率. 与此同时,高流行度内容请求大都在簇内得到响应,也降低了传输跳数. 与次优的 LCD 机制相比,KMHR 机制的路由增益分别提高了 11.30%(如图 4(*a*)中相对缓存空间为 1% 时). 相对缓存空间的增加能提升流行度高的内容占总请求内容的比例,使缓存流行度高的内容带来的效果更加明显,路由增益也逐渐增加.

图 4(*b*)中的瞬时路由增量呈现出与图 3(*b*)相同的走势,即当 *K* 为 10 时,瞬时路由增益最大,与 *K* 为 6、8 的情况相比分别增加了 16.40% 和 4.84%. 当 *K* 值增加时,中心点可缓存的高流行度的内容数量增加,更多的请求可以在簇内得到响应,因此缓存命中逐渐增加,从而使路由增益增加.



(a) 相对缓存空间的影响 (b) 中心点*K*的影响

图4 路由增益的优化

5.2.3 缓存内容数量优化

表 1 体现了五种机制对缓存内容数量的优化效果. KMHR 机制采用 Hash 映射确保流行度高的内容在簇内唯一的缓存,同时以 0.1 的概率缓存流行度低的内

容,极大程度上降低了内容的冗余缓存. 因此 KMHR 机制具有次优的缓存内容数量,仅比 Betw 机制略高. 此外,由于 KMHR 机制中缓存的高流行度内容数量的计算方式为 $M \cdot K \cdot CS_{size}$(当 M 为边缘簇数量,CS_{size} 为内容路由器缓存空间). 因此 KMHR 机制缓存内容的数量随中心点 K 的增加而增加.

表 1　缓存内容数量的优化

	相对缓存空间的影响			中心点 K 的影响		
	0.5%	1%	2%	6	8	10
KMHR	3762	7563	15056	7563	7873	8124
Prob(0.1)	5552	10900	19916	–	–	–
LCE	6200	12360	24620	–	–	–
LCD	6150	12300	24540	–	–	–
Betw	3562	6843	14001	–	–	–

由上述的仿真可以看出,中心点数量 K 的增加将增加请求时间,提升路由增益,增加缓存内容数量. 因而可根据实际网络需求选取适合的 K 值.

6　结论与展望

NDN 特有的网内缓存特性为其他用户访问相同内容提供便利的同时,也带来缓存冗余、缓存效率低的问题. 本文在层次簇网络结构下,结合 on-path 非协作及 off-path 协作缓存路由的特点,提出了基于 K-medoids 的簇内 Hash 路由机制 KMHR. 该机制保证了流行度高的内容在簇内的唯一性,且将大部分的内容请求限制在簇内,降低缓存冗余,提高缓存命中. 仿真结果表明,KMHR 机制的平均请求时间与次优的 LCD 机制相比降低了 7.4%,路由增益比 Betw 机制最多提高了 33.33%,缓存内容数量也比 LCE 机制降低了 39.62%. 后续的研究重点是设计兼顾传输效率及开销的簇间路由机制.

参考文献

[1] Cisco. Cisco Visual Networking Index (VNI):forecast and methodology, 2014 – 2019 [OL]. http://www. cisco. com,2015.

[2] Xylomenos G, Ververidis C, Siris V, et al. A survey of information-centric networking research[J]. IEEE Communications Surveys & Tutorials,2013,16(2):1024 – 1049.

[3] Jacobson V, Smetters D, Thornton J, et al. Networking named content[A]. In Proc. of 2009 ACM CoNEXT conference[C]. Rome, Italy,2009. 1 – 12.

[4] Yi C, Abraham J, Afanasyev A, et al. On the role of routing in named data networking [A]. In Proc. of the 1st[1]nternational Conference on Information-Centric Networking[C]. ACM, Paris, France,2014. 27 – 36.

[5] Laoutaris N, Che H, Stavrakakis I. The lcd interconnection of LRU caches and its analysis[J]. Performance Evaluation,2006,63(7):609 – 634.

[6] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks[A]. In Proc. of the 2nd ACM SIGCOMM workshop on ICN[C]. Helsinki, Finland,2012. 55 – 60.

[7] Chai W, He D, Psaras I, et al. Cache less for more in information-centric networks[A]. In Proc. of the 11th International IFIP TC 6 Networking Conference [C]. Prague, Czech Republic,2012. 27 – 40.

[8] Eum S, Nakauchi K, Murata M, et al. Catt:Potential based routing with content caching for icn[A]. In Proc. of the 2nd ACM SIGCOMM workshop on ICN[C]. Helsinki, Finland,2012. 49 – 54.

[9] Choi H, Yoo J, Chung T, et al. CoRC:Coordinated routing and caching for named data networking[A]. In Proc. of the 10th ACM/IEEE symposium on Architectures for networking and communications systems[C]. New York, USA, 2014. 161 – 172.

[10] Li Z and Simon G. Time-shifted tv in content centric networks:The case for cooperative in-network caching[A]. In Proc. of IEEE International Conference on Communications[C]. Kyoto, Japan,2011. 1 – 6.

[11] Lee M, Cho K, Park K, et al. Scan:Scalable content routing for content-aware networking[A]. In Proc. of IEEE International Conference on Communications[C]. Kyoto, Japan,2011. 1 – 5.

[12] Chatterjee M, Das S K, Turgut D. An on-demand weighted clustering algorithm (wca) for ad hoc networks[A]. In Proc. of IEEE Global Communication Conference[C]. San Francisco, CA,2000. 1697 – 1701.

[13] Kaufman L, Rousseeuw P J. Finding Groups in Data:an Introduction to Cluster Analysis[M]. John Wiley & Sons, 2009.

[14] Hougardy S. The Floyd – Warshall algorithm on graphs with negative cycles[J]. Information Processing Letters, 2010,110(8):279 – 28.

[15] Arora S, Raghavan P, Rao S. Approximation schemes for Euclidean k-medians and related problems[A]. In Proc. of the 30th annual ACM symposium on Theory of computing[C]. 1998. 106 – 113.

[16] Rousseeuw P J, Kaufman L. Finding Groups in Data[M]. Wiley-Interscience, Chichester,1990.

[17] Ross K W. Hash routing for collections of shared web caches[J]. IEEE Network,1997,11(6):37 – 44.

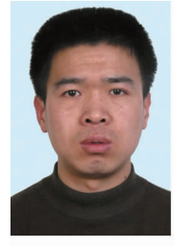
[18] NS-3 based Named Data Networking (NDN) Simulator [EB/OL]. http://ndnsim. net/1. 0/.

[19] Spring N, Mahajan R, Wetherall D. Measuring ISP topologies with Rocketfuel [J]. ACM SIGCOMM Computer Communication Review, 2002, 32(4): 133 – 145.

作者简介

鄢 欢 1986 年 8 月生于河南省潢川县,博士研究生,北京交通大学电子信息工程学院. 主要研究方向为下一代信息网络缓存及传输优化.
E-mail: yanhuanhuan@ bjtu. edu. cn

高德云 1973 年 5 月生于江苏省淮安市,博士,北京交通大学电子信息工程学院教授,博士生导师,主要研究方向为物联网、下一代信息网络理论与关键技术等,主持国家科技重大专项等多项科研项目.
E-mail: gaody@ bjtu. edu. cn



苏 伟 1978 年 10 月生于河北省景县,博士,北京交通大学电子信息工程学院教授,主要研究方向为新一代信息网络关键理论与技术,主持或参与多项国家自然科学基金、863、973 项目.
E-mail: wsu@ bjtu. edu. cn