一种基于逗留时间的新型内容中心网络缓存策略

王国卿[1] 黄 韬[1] 刘 江[1] 陈建亚[2] 刘韵洁[1),2)]

[1)](北京邮电大学网络与交换技术国家重点实验室 北京 100876)

[2)](北京邮电大学北京市网络体系构建与融合重点实验室 北京 100876)

摘 要 内容中心网络(Content-Centric Networking,CCN)作为一种以内容为中心进行路由、缓存的新型未来网络体系架构受到了广泛的关注.在 CCN 中,关键技术问题之一是网络缓存问题,现有方案主要采用 ALWAYS-LRU 缓存策略,然而该策略容易出现相邻节点重复缓存的问题,使得网络整体缓存效率较低.针对这一问题,文中提出了一种基于逗留时间的新型缓存决定策略,设计了一种适用于 CCN 的合作缓存机制.在请求泊松到达的假设下,通过在单个缓存器对 LRU(Least Recently Used)替换策略使用马尔可夫链建模,该文得到了内容在各缓存器平均逗留时间的近似计算公式.数值仿真结果显示,该方案相比传统缓存策略,有效地提升了网络缓存的效率和缓存内容的多样性,进而减少了用户请求服务的总跳数,增加了内容访问的命中率.

关键词 内容中心网络(CCN);逗留时间;LRU;未命中率

中图法分类号 TP393 **DOI 号** 10.3724/SP.J.1016.2015.00472

A New Cache Policy Based on Sojourn Time in Content-Centric Networking

WANG Guo-Qing[1)] HUANG Tao[1)] LIU Jiang[1)] CHEN Jian-Ya[2)] LIU Yun-Jie[1),2)]

[1)](*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876)

[2)](*Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing* 100876)

Abstract As a new Internet architecture, Content-Centric Networking (CCN) use the content name to route and cache, which has been attracted a lot of attention. In CCN, one of the most important issues is in-network caching problem. Although the ALWAYS-LRU cache policy is taken now, this policy leads to the neighbor nodes cache the same contents, which reduces the cache efficiency in the network. Therefore, the decision scheme based on Sojourn Time (ST) is proposed and a new cooperative cache policy is designed in this paper. By modeling the LRU replacement policy at a single node as Markov chain under the assumption of requests following Poisson distribution, the approximate expression of average Sojourn Time is derived. Finally, by comparing with the existing schemes, the numerical results are illustrated to show that the proposed scheme in this paper not only increases the cache efficiency and content diversity, but also decreases the total hops and increases the hit rate.

Keywords CCN; sojourn time; LRU; miss rate

1 引 言

随着互联网技术与应用的飞速发展以及互联网用户的快速增长,基于 TCP/IP 的互联网体系架构逐渐暴露出诸多问题,主要体现在可扩展性差、安全可控性低、移动性支持不足、服务质量难以保证等方面[1].为此,国内外学术界纷纷展开了未来

收稿日期:2012-11-28;最终修改稿收到日期:2014-08-19.本课题得到国家"九七三"重点基础研究发展规划项目基金(2012CB315801)、国家自然科学基金(61300184,61302089)和中央高校基本科研业务费专项资金(2013RC0113)资助.**王国卿**,女,1986 年生,博士,主要研究方向为资源分配、内容中心网络.E-mail:gqwang@bupt.edu.cn.**黄 韬**,男,1980 年生,博士,副教授,主要研究方向为路由与交换、内容中心网络.**刘 江**,男,1983 年生,博士,讲师,主要研究方向为网络虚拟化、内容中心网络.**陈建亚**,男,1951 年生,教授,主要研究领域为无线接入网和下一代网络.**刘韵洁**,男,1943 年生,教授,博士生导师,中国工程院院士,主要研究领域为未来网络、网络融合等.

网络体系架构的研究，从以主机为中心向以内容为中心进行转变，主要涉及的研究项目包括北美的 CCN/NDN①[2-3]、DONA[4] 以及欧盟的 PSIRP/PURSUIT②、NetInf[5] 等等. 这些架构虽然在一些细节上有区别，但是网内节点缓存内容的思想已经达到了共识.

CCN 作为最受关注的未来网络架构之一，其基本思想从被提出之日起，就引发了学术界的一系列的研究热点如内容路由[6]、安全[7]、拥塞控制[8]及网内缓存[9-19]等，尤其是关于 CCN 网内缓存机制的研究受到了学术界的广泛关注. 总结起来，CCN 网内缓存的研究主要体现在两个方面：一方面通过建立数学模型或仿真实验的方法，评估 ICN 系统现有缓存策略的性能[9-13]，其中，文献[9-11]重点研究了 ICN 在线型和树型拓扑下，缓存策略为最近最少使用(LRU)时内容在节点处的命中率表达式，Carofiglio 等人[12]则给出了在 CCNx 原型系统上，分别采用 LRU、最小使用频率(LFU)和生存时间(TTL) 3 种策略的性能评估方案与对比分析结果. Rossi 等人[13]用 ccnSIM 仿真研究了在不同网络拓扑结构下，不同缓存大小的分布对 CCN 网络性能的影响. 另一方面，大量研究致力于提出各种新型缓存策略以提升网络性能，例如清华大学提出的通过跳数感知协议得到利润值的最小利润(Least Benefit，LB)缓存替换算法[14]，以及通过设置内容老化时间以实现驱动流行内容缓存到边缘的算法[15]；Chai 等人[16]提出利用计算缓存器在网络中最短路径上的出现次数来设计内容放置策略的方法；Psaras 等人[17]通过研究中间节点与源节点距离，中间节点到请求节点距离以及中间节点到请求者的剩余路径缓存能力，设计了内容在中间节点缓存概率的放置策略；文献[18]则提出了将大文件分块，利用 Hash 函数将文件块放置到不同缓存节点的方案；Cho 等人[19]提出了随请求次数指数增加逐跳往边缘缓存的内容块数，实行改进的 LCD(Leave Copy Down)策略等.

缓存策略的研究已有多年的历史，完整的缓存策略由两部分组成：缓存决定策略和缓存替换策略. 在 CCN 体系架构提出之前，大多数研究都是针对缓存替换策略展开的[20]，而缓存决定策略方面的研究却很少，主要有随机、LCD(Leave Copy Down)和全存(ALWAYS) 3 种. 随机策略是以一定的概率来决定是否在该节点缓存经过的内容，LCD 策略是让返回的内容仅在第一跳节点缓存，目前，CCN 采用的 ALWAYS 策略对所有经过该缓存器的内容(数据)都缓存. 这 3 种策略的一个共同的缺陷就在于各节点在进行决策时均未综合考虑内容的流行度和缓存容量等因素. 在上述有关 CCN 缓存策略的研究中[14-19]，文献[15-19]都是研究缓存决定策略的，由此可见，CCN 中缓存决定策略的研究已经引起了学术界的重视. 但是在策略的设计方面，这些成果还有一定的改进空间. 文献[15]提出了一种在已存满的节点中若无可替换的内容则不缓存新内容的策略，主要考虑了节点距用户的跳数及内容流行度的因素来设计内容的存活时间，这是一种以替换为主导的决定策略，没有直接对决定策略研究. 文献[16]所提的策略需要计算全网任意两个节点对之间的路径，具有可扩展性差的缺点. 为此，文献[17]提出了在线计算当前路径上各节点的缓存概率的方法，但是该方法仅考虑了节点缓存容量及节点距用户的跳数，没有考虑内容的特点. 文献[18]提出分布式算法保证缓存同一内容不同数据块的节点距离不超过下界的一定的倍数，主要解决了大内容的缓存决策问题. 文献[19]仅利用了内容的流行度，提高流行内容往边缘缓存的收敛速度.

本文主要考虑到要尽量减少跨自治系统(Autonomous Systems，AS)之间的流量，各 AS 应尽可能少地从其他 AS 获取内容，因此在 AS 内实现增加内容多样性的缓存决定策略可以减少 AS 间瓶颈带宽的使用. 研究动机主要是为减少节点对内容的不必要的复制次数，减少域间带宽的使用，同时提高命中率. 本文通过提取出刻画节点对内容竞争力的逗留时间(Sojourn Time，ST)，设计了一种基于逗留时间的缓存决定新策略，逗留时间是指内容从缓存在节点到本次被替换出去的这段时间. 本文先给出了 ST 决定策略的设计方法，将逗留时间加入到兴趣包或数据包中，让节点通过比较内容在相邻节点处的平均逗留时间来决定是否缓存该内容，再结合 LRU 替换策略，得到了完整的 ST-LRU 缓存策略. 随后在特殊的场景下，抽象并建立出马尔可夫链模型，并利用排队论的方法推导出平均逗留时间的近似公式，该公式很好地反应了在有限的存储容量下节点对内容的竞争力. 最后，本文仿真验证了新策略在复制次数，总跳数及命中率上的优势.

① Named Data Networking. http://named-data.net/wp-content/uploads/2014/04/tr-ndn-0019-ndn.pdf, 2014,4,10

② Conceptual Architecture: Principles, patterns and sub-components descriptions. http://fp7pursuit.ipower.com/PursuitWeb/wp-content/uploads/2011/06/INFSO-ICT-257217_PURSUIT_D2.2_Conceptual_Architecture_Principles_patterns_and_sub-components_descriptions.pdf, 2011,5,16

2 基于逗留时间的缓存策略

2.1 CCN 系统模型描述

在 CCN 中，内容被分成数个大小相同的数据块，数据块被唯一命名，被永久地存储在一个或多个服务器中. CCN 的数据传输主要由数据包(Data Packet)和兴趣包(Interest Packet)完成，用户利用接收者驱动的传输协议发送兴趣包，基于名字的路由协议确保了请求被路由到合适的缓存器，每个中间节点都会跟踪该节点未命中的请求，以便将被请求的数据原路返回到接收者以及让途经的节点缓存数据，此外，中间节点还将请求聚合，即当第一个请求还未被满足时，避免转发对该同一个数据的其他请求. 数据可能来自服务器或来自任何被命中的缓存器，即路径上暂时保存内容副本的节点，缓存器采用 LRU 替换策略.

LRU 机制的核心工作机理是当需要缓存内容 k 时，若内容 k 已经缓存在 CS 中，则将内容 k 移到最上面，原来在内容 k 上面的内容下移一位；若 CS 中没有内容 k，则将内容 k 缓存在最上面，原来在 CS 中的每个内容下移一位，原来在 CS 中最下面的内容被替换出去.

由于内容的总大小远大于单个节点的缓存容量，因此被缓存的内容经常会被新到的内容替换出去. 因此，内容的平均逗留时间是一个有限的实数.

从 ST 和 LRU 的定义不难看出，ST 值应服从如下规则：(1) 缓存器容量越大，在该缓存器的内容的 ST 值也越大；(2) 内容流行度越高，ST 值也会越大. 显然，通过比较内容在相邻缓存器的逗留时间大小，把内容缓存在逗留时间较长(即 ST 值较大)的节点能够有效减少内容的复制次数，增加 AS 域内的内容多样性.

下面用一个简单的场景来比较 ST-LRU 机制较 ALWAYS-LRU 机制的优势，如图 1 所示. 用 $ST_k(i)$ 表示内容 k 在节点 i 处的平均逗留时间，假设一个由两个路由器 a 和 R_2 在 R_1 中的平均逗留时间大于内容 R_2 可以与其他 AS 域通信. 通过对内容 a 及其他内容在 R_1 和 R_2 的到达率等因素的分析，已知内容 a 在 R_1 中的平均逗留时间大于内容 a 在 R_2 中的平均逗留时间，即 $ST_a(1)>ST_a(2)$.

假设 R_1 处有一个内容 b 的请求，内容 b 返回之后，b 与 R_2 中都缓存了内容 b，然后 R_1 处又有一个内容 a 的请求，返回的内容 a 在 ALWAYS 策略下在 R_1 和 R_2 中都缓存；而在 ST 策略下，由于 $ST_a(1)>ST_a(2)$，内容 a 仅缓存在 R_1 中. 显然，ST 机制有效地避免了相邻节点缓存相同内容的问题，有效地增加了 AS 域内内容存储的多样性.

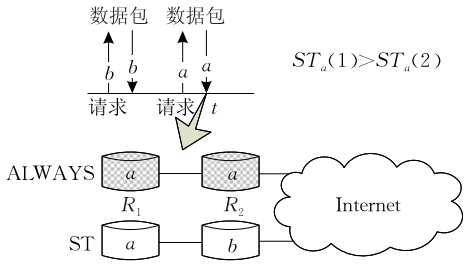


图 1 ALWAYS 与 ST 策略对比示意图

2.2 ST 缓存决定策略在 CCN 中的设计

CCN 路由器有 3 个数据结构表：内容存储表(Content Store，CS)、跟踪兴趣包表(Pending Interest Table，PIT)和转发信息表(Forwarding Information Base，FIB). 其中，CS 表用来缓存所经过的内容，PIT 表用来记录那些刚被请求但还没有得到回应的内容条目及其转入接口，FIB 表则记录内容名对应的下一跳接口. 未命中的兴趣包(CS 中没有缓存被请求的内容)和它的接口被记录在 PIT 表中，并且通过查找 FIB 表将该兴趣包转发到下一跳，返回的数据包通过查 PIT 按兴趣包原路返回到请求者. ALWAYS 缓存决定策略是将所有经过路由器的 PIT 中有记录的内容缓存到 CS 中，而 ST 机制设计的核心目标是让内容缓存到合适的路由器中. 本文将 ST_k 加入到内容 k 的兴趣包和数据包中，并将该值通告给邻居节点.

节点 i 接收到从节点 j 处发来的内容 k 的兴趣包，节点 i 先检查其 CS 中是否存有内容 k，若有则直接返回内容 k；若无，则查询 PIT 表：若 PIT 表中有内容 k 这个条目，则在该条目中加入接口 j 及 $ST_k(j)$，并且丢弃该兴趣包；若 PIT 表中没有内容 k 的条目，则创建一个新的条目，记录该内容前缀和接口 j 及 $ST_k(j)$，随后将 $ST_k(i)$ 覆盖内容 k 的请求包中的 $ST_k(j)$，并按照 FIB 表的接口记录转发到下一跳. 于是，改进后的 PIT 表设计结构如图 2 所示.

PIT 表	
前缀	请求接口
…	…
k $ST_k(i)$	j $ST_k(j)$；l $ST_k(l)$；…
…	…

图 2 基于 ST 机制下的 CCN 网络 PIT 表设计结构

当内容 k 从节点 j 返回到节点 i 时，首先查询 PIT 表中是否有内容 k 的条目，若无，则直接丢弃该数据包，若有，则将该数据包中携带的 $ST_k(j)$ 与 PIT 表中内容 k 条目中记录的 ST_k 比较，当且仅当 $ST_k(i)$ 大于其他几个 ST_k 值时，才将内容 k 缓存到节点 i 的 CS 中. 然后将 $ST_k(i)$ 加入到数据包并覆盖数据包中的 $ST_k(j)$，并将该数据包转发到 PIT 表记录的所有接口，具体算法流程如图 3 所示.

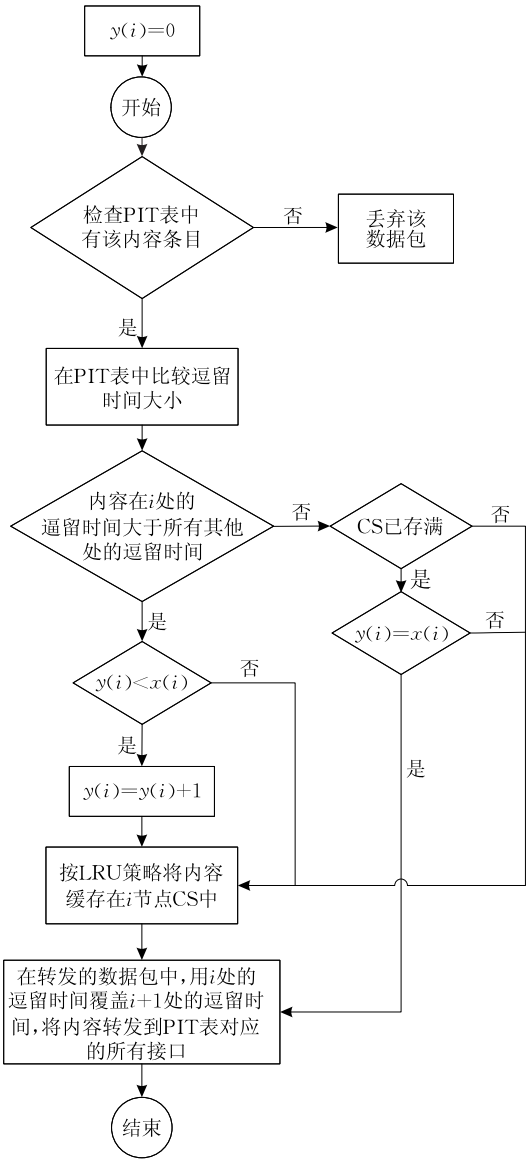


图 3　从节点 j 返回的内容 k 在节点 i 的算法流程图

在图 3 中，$y(i)$ 用于记录缓存器 i 中按照逗留时间挑选出的缓存内容个数，$x(i)$ 表示缓存器 i 的缓存容量，有 $y(i)\leq x(i)$. 在缓存器 i 还未接收到任何内容之前，缓存器 i 中没有缓存内容，故 $y(i)=0$，开始运行之后，节点 i 接收到内容，先查找 PIT 表，如果 PIT 表中无该内容条目记录，说明该内容已被返回或者出错，因此丢弃该内容；如果 PIT 表中有该内容条目，则把 PIT 中记录的该内容条目对应的逗留时间与数据包携带的上游节点的逗留时间比较，若缓存器 i 处的逗留时间大于其他所有逗留时间，则存入节点 i，同时检验按逗留时间大小缓存的内容个数 $y(i)$ 是否已将缓存器存满，即判断是否 $y(i)<x(i)$，目的是要更新 $y(i)$；若缓存器 i 处的逗留时间不大于其他所有逗留时间，则判断 CS 是否已经存满，若已经存满且 $y(i)=x(i)$，则不缓存该内容；若 CS 未存满，则缓存该内容，以避免浪费缓存器的空间.

从该算法可以看出，ST-LRU 机制会让内容缓存在平均逗留时间较大的节点中，相邻节点不会缓存相同的内容，从而达到增加多样性、减少复制次数和提高命中率的目的. 在两种极端情形下，节点 i 将决定缓存所有经过的内容，ST 机制将与 ALWAYS 机制有一样的效果，这两种情形是：(1) 仅有小于 $x(i)$ 种内容的 $ST_k(i)$ 小于邻居处的 ST_k（即 $y(i)<x(i)$）；(2) $ST_k(i)$ 恒大于邻居处的 ST_k（$\forall k$）. 这是因为当 $y(i)<x(i)$ 时，新到的内容总是会被缓存；当节点 i 处所有内容的逗留时间都大于其相邻节点处的相应内容的逗留时间时，新到的内容也将全被缓存.

2.3　ST-LRU 策略举例

为了便于分析 ST-LRU 策略，先给出 ST 值的近似计算公式如下：

$$ST_k(i)=\frac{x(i)}{\mu_k(i)}-\frac{1}{\lambda_k(i)}\left(1-\left(\frac{\mu_k(i)}{\lambda_k(i)+\mu_k(i)}\right)^{x(i)}\right) \quad (1)$$

其中，$\mu_k(i)=\lambda(i)-\lambda_k(i)$. 其详细推导将在第 3 节中给出. 从式(1)可以看出，ST 值与流行度和缓存容量的变化关系符合 2.1 节所述规则.

下面用一个简单的场景来描述 ST-LRU 机制的整个流程，并与 ALWAYS-LRU 策略比较. 假设由两个路由器 R_1 和 R_2 构成的一个 AS 域，缓存容量分别为 2 和 1 个内容，即 $x(1)=2$，$x(2)=1$. 每个节点接收到内容 a，b 和 c 的请求率如图 4 给出. 这样，通过式(1)可以算出内容的 ST 值如表 1 所示.

表 1　内容在节点的平均逗留时间　（单位：s）

内容	节点 1	节点 2
a	5/12	1/30
b	2/9	1/12
c	17/180	1/6

本例仅考虑与 a 的请求到达 R_1，由于 R_1 中没有内容 a 的请求到达 R_1，由于 R_1 中没有内容 a，请求被转发到 R_2，R_2 中也没有内容 a，从而，请求被转

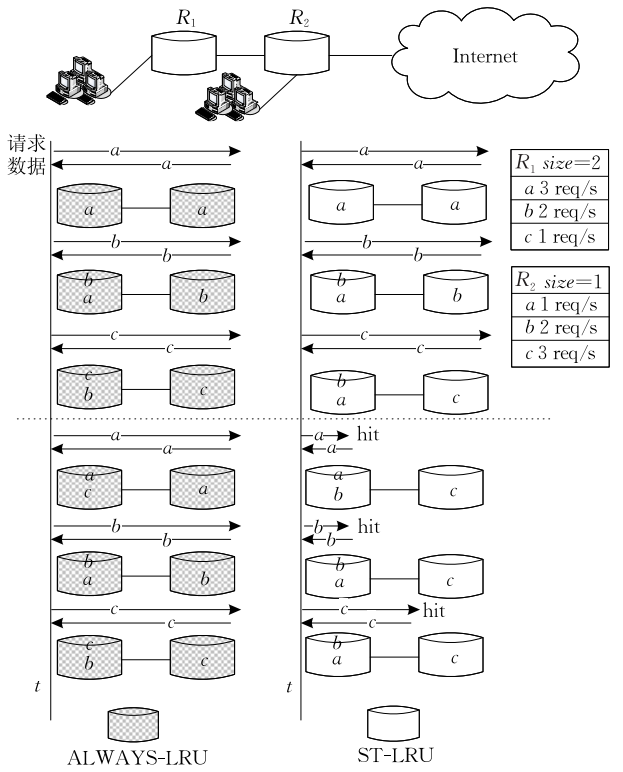


图 4 ALWAYS-LRU 与 ST-LRU 缓存机制对比

发到其他 AS 域，返回的内容 a 先到达 R_2，比较 $ST_a(2)$ 与 $ST_a(1)$，有 $ST_a(2)<ST_a(1)$，然后查看 CS 是否已满，若否，将内容 a 缓存到 CS 中，然后转发至 R_1，R_1 采用同样的算法进行实现（如图 3 所示），即首先将内容 a 缓存到 CS 中，由于 $ST_a(1)>ST_a(2)$ 且 $y(1)<x(1)$，$y(1)$ 自增 1. 接下来内容的请求过程与上述过程类似，不再赘述.

初始化结束后（虚线以下），R_1 又收到内容 a 的请求，ST-LRU 机制下 R_1 就可以直接返回内容 a（即命中），而在 ALWAYS-LRU 机制下，该请求在本地未命中，被转发到其他 AS 域，至少需要两跳才可以命中. 而且，在虚线以下，ST-LRU 策略几乎稳定，即 R_1 缓存内容 a 和 b，R_2 缓存内容 c，内容 a，b 和 c 都不会再被复制，而 ALWAYS-LRU 机制会一直复制. 显然，ST-LRU 的新策略有效地提高了命中率，减少了跳数及不必要的内容复制次数.

3 ST-LRU 方案的性能分析

3.1 CCN 数学模型及假设

本文首先建立一个具有 N 个节点（CCN 路由器）的 CCN 网络模型，这 N 个节点在一个 AS 域中，网络中共有 M 种不同的内容，这些内容大小一样[10-12]，永久地存储在该 AS 域外的一个或多个服务器中，即所有内容一定可以从该 AS 域外获得. 假设节点 i 可以缓存 $x(i)$ 个内容，$x(i)$ 远小于 M. 节点接收到的请求过程是由本节点的用户请求序列和邻节点的未命中（miss）序列构成的. 假设节点 i 从用户处接收到的请求服从参数为 $\lambda^u(i)$ 的泊松分布，根据随机服务系统原理，服从泊松分布的输入流的输出也服从泊松分布，因此，邻节点未命中的请求序列也服从参数为 $\eta(i)$ 的泊松分布. 因此，节点 i 的输入过程服从参数为 $\lambda(i)=\lambda^u(i)+\eta(i)$ 的泊松分布.

本文涉及到的主要符号如表 2 所示.

表 2 本文主要符号及意义

符号	意义
N	节点（CCN 路由器）个数
M	内容种类数
$x(i)$	节点 i 缓存容量（个内容）
$\lambda^u(i)$	节点 i 收到用户的请求率（个/s）
$\eta(i)$	节点 i 收到下游节点 miss 的请求率（个/s）
$\lambda(i)$	节点 i 收到的总请求率
$\lambda'(i)$	节点 i 决定存的内容的总请求率
$\lambda_k(i)$	节点 i 收到内容 k 的请求率
$ST_k(i)$	内容 k 在节点 i 的平均逗留时间

3.2 平均逗留时间的计算公式

本文在分析过程中采用了经典的 LRU 替换策略，在进行平均逗留时间的设计之前，我们首先建立一个 LRU 策略的数学模型.

首先，因为内容 k 所在位置都只与前一个位置有关，与再之前所处的位置独立，又因为请求是泊松到达的，内容 k 两个位置间的转移仅发生在请求的到达时刻，泊松到达仅与时间间距相关，因此，我们将 LRU 的工作过程抽象成为一条齐次马氏链[9]，链的状态就是内容 k 在 CS 中的位置：状态 0 表示内容 k 不在 CS 中，状态 1 表示内容 k 在 CS 的最上面，状态 l 表示内容 k 在 CS 的第 l 位置. 从状态 l（$l=1,2,\cdots,x$）转到状态（$l+1$）的转移率为 μ_k，其中 $(x+1)$ 表示状态 0，每个状态 l 都可以转移到状态 1（$l\neq 1$），转移率为 λ_k（内容 k 的到达率）（如图 5 所示）. 于是，这条马氏链的转移率矩阵可表示为

$$\boldsymbol{Q}=\begin{pmatrix} -\lambda_k & \lambda_k & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\mu_k & \mu_k & 0 & \cdots & 0 & 0 \\ 0 & \lambda_k & -\beta & & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \lambda_k & 0 & 0 & \cdots & -\beta & \mu_k \\ \mu_k & \lambda_k & 0 & 0 & \cdots & 0 & -\beta \end{pmatrix},$$

这里，$\beta=\lambda_k+\mu_k$.

对于满足 $\lambda_k/\mu_k<1$ 的内容 k，它对应的上述马

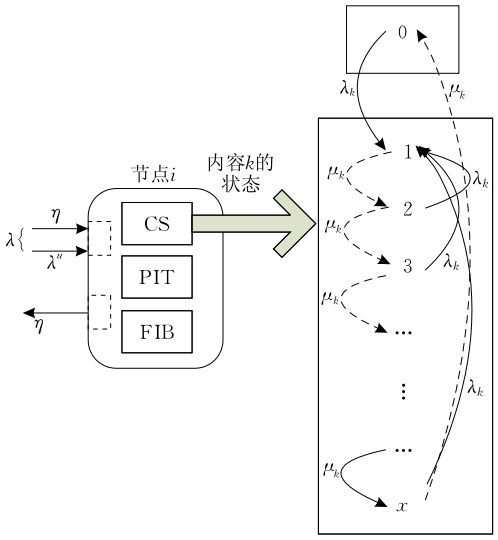


图 5　缓存器中内容 k 的状态转移图

氏链是非周期、不可约、正常返的，因此，一定存在稳态概率，可通过解下面的稳态方程来求解：

$$\begin{cases}\boldsymbol{\pi}_k \boldsymbol{Q}=0\\ \boldsymbol{\pi}_k \boldsymbol{e}=1\end{cases} \tag{2}$$

这里，$\boldsymbol{\pi}_k$ 是一个 $(x+1)$ 维的列向量，且 $\boldsymbol{e}=(1,1,\cdots,1)^{\mathrm{T}}$ 是一个 $(x+1)$ 维的列向量. 则有

$$\pi_{kl}=\frac{\lambda_k}{\mu_k}\left(\frac{\mu_k}{\mu_k+\lambda_k}\right)^l,\ l=1,2,\cdots,x \tag{3}$$

$$\pi_{k0}=\left(\frac{\mu_k}{\mu_k+\lambda_k}\right)^x \tag{4}$$

其中，π_{kl} 表示稳态时内容 k 处在状态 l 的概率. 由此，可以推导出内容 k 在缓存器中平均逗留时间的公式.

命题 1.　缓存器收到的内容请求服从参数为 λ' 的泊松到达，内容 k 的请求服从参数为 λ_k 的泊松过程 $(k=1,2,\cdots,M)$，内容具有相同的大小，缓存容量为 x 个内容，则内容 k 在缓存器中的平均逗留时间近似公式可表示为

$$ST_k=\frac{x}{\mu_k'}-\frac{1}{\lambda_k}\left(1-\left(\frac{\mu_k'}{\lambda_k+\mu_k'}\right)^x\right) \tag{5}$$

其中，$\mu_k'=\lambda'-\lambda_k$.

证明.　若内容 k 不在 0 状态，则可以将除内容 k 以外决定缓存的内容分成两类：(a) 未被缓存的内容及存在内容 k 下面的内容；(b) 存在内容 k 上面的内容. 而如果内容 k 在 0 状态，则不需要分类. 显然，只有(a)类内容及内容 k 的到达才会改变内容 k 的状态，而(b)类内容的到达不会改变内容 k 的状态.

令 ω_k 为内容 k 在缓存器中的逗留时间，$Pr_j\{\omega_k>t\}$ 表示在内容 k 处于状态 j 的条件下，内容 k 的逗留时间大于 t 的概率，则有

$$Pr\{\omega_k>t\}=\sum_{j=1}^{x}\pi_{kj}Pr_j\{\omega_k>t\} \tag{6}$$

CCN 网络中请求到达序列与一般的排队系统到达序列的不同之处在于：对同一内容的请求可能在前一个请求还未被满足之前重复到达，这样该内容就会由(a)类内容变成(b)类内容. 在本模型中，当内容 k 在缓存器的第 j 个位置等待被替换出去的时候，在下一个内容 k 的请求到达之前的时间 t 内，当且仅当有不少于 $(x-j+1)$ 个不同的(a)类内容请求到达，内容 k 才会被替换出去. 即在 t 这段时间内，当且仅当到达的(a)类内容请求数小于 $(x-j+1)$ 时，逗留时间才会大于 t，因此有

$$Pr_j\{\omega_k>t\}=\sum_{i=0}^{x-j}\mathrm{e}^{-\mu_k t}\frac{(\mu_k t)^i}{i!} \tag{7}$$

将式(7)代入式(6)中，得到

$$\begin{aligned}Pr\{\omega_k>t\}&=\sum_{j=1}^{x}\pi_{kj}\sum_{i=0}^{x-j}\mathrm{e}^{-\mu_k t}\frac{(\mu_k t)^i}{i!}\\&=\mathrm{e}^{-\mu_k t}\left[\sum_{i=0}^{x-1}\frac{(\mu_k t)^i}{i!}\left(1-\left(\frac{\mu_k}{\lambda_k+\mu_k}\right)^{x-i}\right)\right]\end{aligned} \tag{8}$$

从而得到内容 k 的平均逗留时间：

$$\begin{aligned}ST_k&=E\omega_k\\&=\int_0^{\infty}td(1-Pr\{\omega_k>t\})\\&=\frac{x}{\mu_k}-\frac{1}{\lambda_k}\left(1-\left(\frac{\mu_k}{\lambda_k+\mu_k}\right)^x\right)\end{aligned} \tag{9}$$

从式(9)可以发现，等号右边的表达式与 x 和 λ_k 正相关，与 μ_k 负相关，与我们预期的 ST_k 与这三者之间的关系一致.

下面进一步分析内容 k 状态的向下转移率 μ_k 的取值. 由上文可知，当内容 k 处于非零状态时，只有(a)类内容的到达才会让内容 k 的状态值加 1，而(b)类内容的到达不会改变内容 k 的状态. 因此，得到

$$\mu_k^{\mathrm{upper}}=\lambda'-\lambda_k \tag{10}$$

$$\mu_k^{\mathrm{lower}}=\begin{cases}\lambda'-\sum_{j=1}^{x}\lambda_j, & k<x\\ \lambda'-\lambda_k-\sum_{j=1}^{x-1}\lambda_j, & k\geq x\end{cases} \tag{11}$$

当内容 k 处在状态 1 时，μ_k 取得最大值，即式(10)；当内容 k 处在状态 x，并且存在它上面的内容是最流行的 $(x-1)$ 个内容时，μ_k 取得最小值，即式(11).

由于 π_{k0} 是 μ_k 的不减函数，于是可以得到内容 k 的未命中概率区间为

$$\left[\left(\frac{\mu_k^{\text{lower}}}{\mu_k^{\text{lower}}+\lambda_k}\right)^x,\left(\frac{\mu_k^{\text{upper}}}{\mu_k^{\text{upper}}+\lambda_k}\right)^x\right].$$

式(5)的右侧为 μ_k 的不增函数，当内容请求的分布比较均匀(即在时间 t 内到达两个或以上相同内容的请求几乎不可能)，并且在 $x(i)$ 远小于 M 的情况下，(b)类内容相对于(a)类内容很少. 因此，可以推得 ST_k 的近似计算公式：

$$ST_k=\frac{x}{\mu_k'}-\frac{1}{\lambda_k}\left(1-\left(\frac{\mu_k'}{\lambda_k+\mu_k}\right)^x\right) \tag{12}$$

其中 $\mu_k'=\mu_k^{\text{upper}}=\lambda'-\lambda_k$. 证毕.

3.3 ST-LRU 方案的定性分析

从上述命题的证明过程可知，当内容请求分布让短时间内重复请求同一个内容的概率很小时，式(5)更有效，即若请求服从 Zipf 分布，则非负参数 α 越小越好. 此外，由于式(5)中 μ_k 的取值忽略了缓存在内容 k 上面的内容，即(b)类内容，因此缓存器容量 x 也是越小越好(x 越小，缓存的内容就越少，从而缓存在内容 k 上面的内容也越少).

由上节分析，可将传统 ALWAYS-LRU 机制的未命中率表示为

$$m_k(i)=\lambda_k(i)\left(\frac{\mu_k(i)}{\mu_k(i)+\lambda_k(i)}\right)^{x(i)},\ \mu_k(i)=\lambda(i)-\lambda_k(i) \tag{13}$$

同理，可将 ST-LRU 机制的未命中率表示为

$$m_k(i)=\lambda_k(i)\left(\frac{\mu_k'(i)}{\mu_k'(i)+\lambda_k(i)}\right)^{x(i)}I_k(i)+\lambda_k(i)(1-I_k(i)),$$

$$\mu_k'(i)=\lambda'(i)-\lambda_k(i),\ I_k(i)=\begin{cases}0, & \text{节点 } i \text{ 不存内容 } k\\ 1, & \text{节点 } i \text{ 存内容 } k\end{cases} \tag{14}$$

由 ST-LRU 策略的算法及式(13)和式(14)可以得出新策略设计的合理性如下：(1)减少了内容的复制次数，就是通过让平均逗留时间长的内容在缓存器中停留更长时间，不复制平均逗留时间较短的内容来实现的. (2)减小了总的未命中率($=\sum_k m_k(i)$)，从而减少了总跳数，这主要有两方面的原因：一方面，新策略减小了决定缓存的内容个数，即 $\mu_k'\leq\mu_k$，由于未命中概率 π_{k0} 随着 μ_k 的减小而减小，从而决定缓存的内容未命中率有所减小；另一方面，不决定缓存内容的未命中概率虽然为 1，但是不决定缓存的内容的请求率也很小，因此有效减少了系统总的未命中率. (3)增加了 AS 域内的内容多样性，从而减少了 AS 域间的流量传输. 通过相邻节点之间的比较决定是否缓存内容，有效避免了相邻节点缓存相同的内容，提高了系统的整体缓存效率.

在该策略的设计中，兴趣包和数据包在原包的基础上又增加了一段空间携带该内容在前一个节点处的逗留时间数值，并且给节点增加了读取与比较相邻节点逗留时间的计算代价. 该策略在传输过程中没有增加新的包，在读取的过程中也只是在原来读取内容前缀及接口的基础上多读取了一个数据——逗留时间. 这些代价远远小于将相邻两个节点的缓存空间充分利用所节省的存储成本.

4 仿真结果与分析

4.1 仿真环境设置

本文考虑一个具有 10 个节点的 CCN 网络模型($N=10$)，CS 初始为空，$M=1000$ 个相同大小的内容存储在服务器中. 用户和其他节点可以发送请求到他们相邻的上游节点. 用户发送到节点的请求服从泊松分布($\lambda^u=5$ 个/s)，内容请求服从参数为 $\alpha(i)=\alpha, i=1,2,\cdots,10$ 的 Zipf 分布，节点 i 的请求概率设为 $q_k(i)=c(i)/k^{\alpha(i)}$，其中 $1/c(i)=\sum_k 1/k^{\alpha(i)}$ 为归一化参数(即内容的流行度从大到小排序，最流行的内容标识为 1). 由于 CCN 目前尚未在真实网络中部署，本文仿真是通过理论分析的数值结果验证[10-12,15]，与实际结果之间可能出现偏差.

在 CCN 中，无论采用何种网络拓扑结构，其最基本的场景都是，每个节点都接收到两种请求流：一种是与其直接相连的用户的请求，另一种是其他节点未命中的请求流之和. 在网状拓扑结构中，每个节点收到的请求流也是上述两种，区别仅仅是 $\eta(i)$ 数值上的变化. 因此为便于问题的分析(不考虑路由策略)，同时不影响通用性，本文仿真任意网络拓扑结构中的一条路由路径，同时假设每个节点(节点 1 除外)都接收到两种请求流(如图 6 所示).

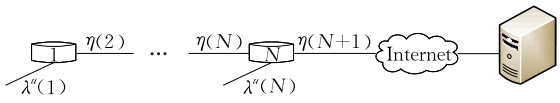


图 6 网络拓扑结构中的一条路由路径

需要说明的是，第 2 节所述的算法完整地比较了上游与下游的 ST 值，本文将此算法称作 ST_{both} 机制，很容易得到两种简化的机制——ST_{up} 和 ST_{down} 机制：ST_{up} 即仅与上游 ST 值比较，ST 值仅需添加到数据包中，节点 i 决定缓存内容 k 当且仅当 $ST_k(i)\geq ST_k(i+1), i=1,2,\cdots,9$，这里假设第 10 个节点决定缓存第 9 个节点所有不决定存的内容；

ST_{down}即仅与下游 ST 值比较，ST 值仅需添加到兴趣包中，节点 i 缓存内容 k 当且仅当 $ST_k(i)\geq ST_k(i-1)$，$i=2,3,\cdots,10$，这里假设第 1 个节点决定缓存第 2 个节点所有不决定存的内容.

ALWAYS-LRU 策略节点 i 获取内容 k 所需跳数可表示为

$$hop_k(i)=\lambda_k(i)\Big[(1-\pi_{k0}(i))+\sum_{j=i+1}^{N}(j-i+1)(1-\pi_{k0}(j))\prod_{l=i}^{j-1}\pi_{k0}(l)+10N\prod_{l=i}^{N}\pi_{k0}(l)\Big] \tag{15}$$

其中，$(1-\pi_{k0}(i))$表示在节点 i 命中的概率；$(1-\pi_{k0}(j))\prod_{l=i}^{j-1}\pi_{k0}(l)$表示从节点 i 到节点 $j-1$ 均未命中、节点 j 命中的概率；$10N\prod_{l=i}^{N}\pi_{k0}(l)$表示若从节点 i 到节点 N 均未命中，则需要 $10N$ 跳才能命中的概率. 该假设的目的是将 AS 域间带宽的使用程度间接转化为跳数的多少，这样，若内容需要从其他域获取，则所需跳数剧增；反之，若节点获取内容的总跳数较大，则说明大部分内容请求在本 AS 域中未命中，需从其他 AS 域获取.

从式(15)可以看出，本文只考虑了一个方向的路由路径，即从节点 i 往节点 $i+1$ 路由的场景，没能更好地利用 AS 域中缓存的内容，特别是节点 N，它接收到的请求只能命中节点 N 缓存的内容，其余的请求全未命中而被转发到其他 AS 域(或服务器)获取内容，从而大大地增加了总跳数.

与式(15)类似，ST-LRU 策略中节点 i 获取内容 k 所需跳数可表示为

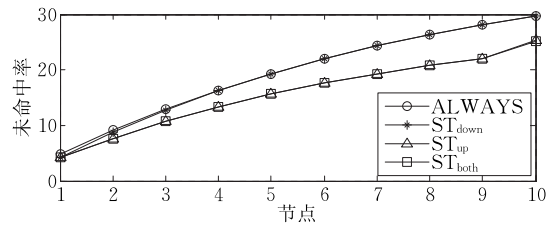
$$hop'_k(i)=\lambda_k(i)\Big[(1-\pi'_{k0}(i))+\sum_{j=i+1}^{N}(j-i+1)(1-\pi'_{k0}(j))\prod_{l=i}^{j-1}\pi'_{k0}(l)+10N\prod_{l=i}^{N}\pi'_{k0}(l)\Big] \tag{16}$$

其中，$\pi'_{k0}(i)=\left(\frac{\mu'_k(i)}{\mu'_k(i)+\lambda_k(i)}\right)^{x(i)}I_k(i)+(1-I_k(i))$.

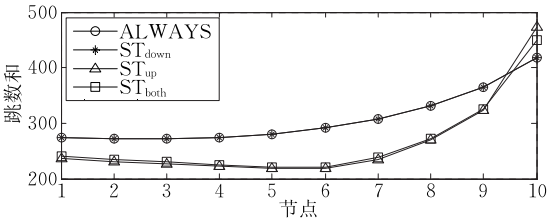
4.2　仿真结果分析

4.2.1　ST-LRU 与 ALWAYS-LRU 机制的比较

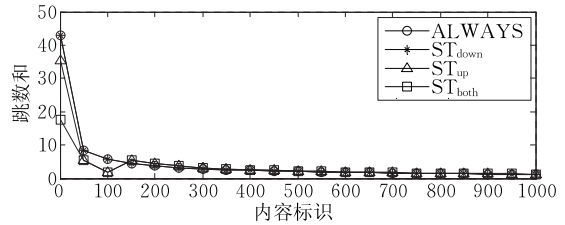
图 7 给出了随缓存器大小在递增情况下几种缓存策略的性能比较，图 7(a)为请求分别到达所有节点处的未命中率之和，图 7(b)为各个节点取得内容所需的总跳数，图 7(c)为在 50 个请求中(每个节点从用户处接收到 5 个请求)，每种请求获取内容所需的总跳数. 可以看出，ST_{both}和ST_{up}策略均优于ALWAYS策略，ST_{down}与 ALWAYS 策略近似，这是因为在缓存容量为[5,10,15,20,25,30,35,40,45,50]时，由平均逗留时间式(5)可知，几乎所有内容在本节点的 ST 值都大于下游节点的 ST 值，即 ST_{down}几乎所有内容都存，从而 ST_{down}与 ALWAYS 策略几乎一样. 图 7(b)中虽然第 10 个节点处出现 ST_{both}和 ST_{up}策略所需跳数大于 ALWAYS 与 ST_{down}策略，但是由图 7(c)可知，ST_{both}和 ST_{up}策略下所有节点对同一内容的请求所需总跳数是小于 ALWAYS 和 ST_{down}的.



(a) 各策略在各节点处的未命中率比较



(b) 各策略在各节点处的跳数和比较

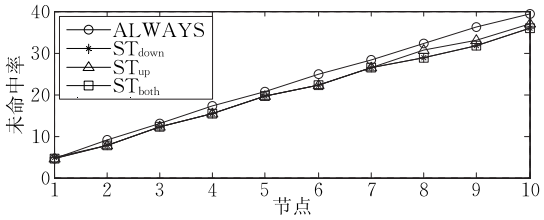


(c) 各策略获取各内容的跳数和比较

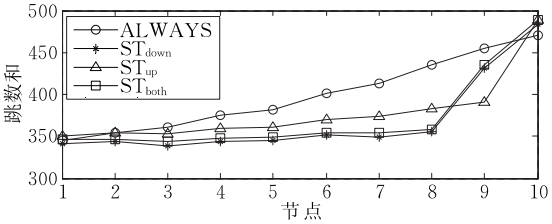
图 7　缓存容量为[5:5:50]时的性能比较

因为本文的仿真只考虑了往节点 10 方向路由的场景，没能更好地利用 AS 域中缓存的内容，特别是节点 10，它只利用了自己缓存的内容，未命中的请求全都到其他 AS 域取. 所以，节点 10 处 ST_{both}和 ST_{up}策略所需跳数将大于 ALWAYS 与 ST_{down}策略所需跳数.

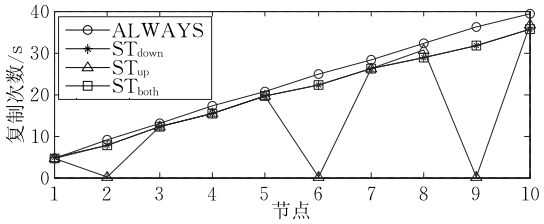
图 8 则给出了 10 个节点缓存容量分别为[6,15]之间的整数随机排列时，4 种策略在未命中率、总跳数和单位时间复制次数的比较. 仿真结果显示，ST_{both}，ST_{up}和 ST_{down}策略在这 3 方面的性能均优于传统的 ALWAYS 策略.



(a) 各策略在各节点处的未命中率比较



(b) 各策略在各节点处的跳数和比较



(c) 各策略在各节点处的每秒复制次数

图 8 缓存容量为[13,9,14,6,15,8,12,10,7,11]时的性能比较

4.2.2 平均逗留时间计算公式的有效性分析

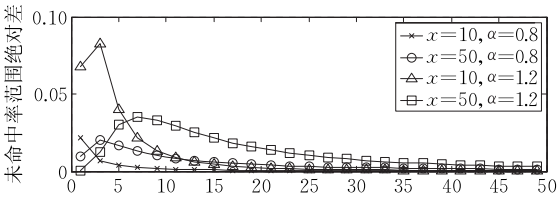
本文的 3.2 节从原理上简要阐述了平均逗留时间式(5)的适用范围,本节将从未命中率和 ST 值取值范围的绝对差来定量分析式(5)的适用范围. 取值范围的绝对差越小,则式(5)在该场景下越有效(因为式(5)是取 ST 值变化范围间的某值来近似平均逗留时间). 其中,未命中率的变化范围绝对差可表示为

$$\lambda_k\left[\left(\frac{\mu_k^{\text{upper}}}{\mu_k^{\text{upper}}+\lambda_k}\right)^x-\left(\frac{\mu_k^{\text{lower}}}{\mu_k^{\text{lower}}+\lambda_k}\right)^x\right] \tag{17}$$

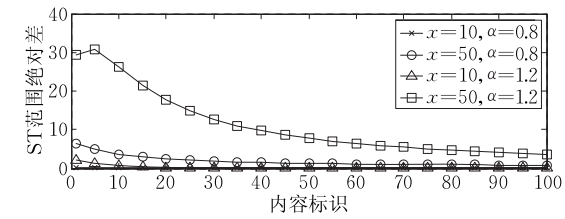
ST 值的变化范围绝对差可表示为

$$\frac{x}{\mu_k^{\text{lower}}}-\frac{1}{\lambda_k}\left(1-\left(\frac{\mu_k^{\text{lower}}}{\lambda_k+\mu_k^{\text{lower}}}\right)^x\right)-\frac{x}{\mu_k^{\text{upper}}}+\frac{1}{\lambda_k}\left(1-\left(\frac{\mu_k^{\text{upper}}}{\lambda_k+\mu_k^{\text{upper}}}\right)^x\right) \tag{18}$$

图 9(a)中,当 $\alpha>1$ 时,前 5 个最流行的内容的未命中率的变化范围绝对差都在 $x=10$ 下小于 $x=50$ 的情况;当 $\alpha<1$ 时,除了第 1 个最流行的内容外,都有未命中率在 $x=10$ 下小于 $x=50$ 的情况. 而在 x 相同的情况下,$\alpha<1$ 的未命中率变化范围绝对差小于 $\alpha>1$. 因此,由图 9(a)我们得到 $\alpha<1, x=10$ 是最佳的参数取值. 图 9(b)也显示 $\alpha<1, x=10$ 是最佳的参数取值. 该结论与 3.2 节的分析相符.



(a) 前50种最流行内容的未命中范围绝对差在不同参数值下的比较



(b) 前100种最流行内容的ST范围绝对差在不同参数值下的比较

图 9 未命中率与 ST 值的变化范围绝对差

5 小 结

本文提出了一个新的基于逗留时间的缓存决定策略,该策略的目标是缓存决策时尽可能地选择逗留时间较长的内容,从而有效地减少了内容的复制次数,增加了域内命中次数和减少域间带宽使用. 本策略充分地考虑了缓存器容量及内容的流行度,并适用于所有网络拓扑结构. 然而,该策略也带来了一定的处理负担,即节点需要读取与更新比较兴趣包或数据包携带的平均逗留时间的信息. 本文在系统实现时采用了在数据包或兴趣包中添加一个逐跳更新数的方法来达到了交互的目的. 事实上,对于 CCN 这种分布式网络架构,节点间信息交互是很自然的,同时也可以根据实际需求合理控制节点之间的交互范围. 在网络鲁棒性方面,由于逗留时间会随着网络状况的变化而变化,因此在动态变化的网络中,可以增加设计逗留时间的更新,按照不断更新的逗留时间来动态地选择所缓存的内容,从而保证网络的鲁棒性.

本文的策略在缓存容量较小、请求分布较均匀的场景下更有效,后续我们将通过给每个内容计时的方法,估算内容的平均逗留时间,这样该策略就能够应用到任何场景下了. 此外,我们将在后续 CCN 网内缓存机制的研究中,尝试考虑拓扑结构、带宽等因素,从而进一步提升缓存机制的效率和命中率.

参 考 文 献

[1] Lin Chuang, Jia Zi-Xiao, Meng Kun. Research on adaptive

future Internet architecture. Chinese Journal of Computers, 2012, 35(6): 1077-1093(in Chinese)
(林闯, 贾子骁, 孟坤. 自适应的未来网络体系架构. 计算机学报, 2012, 35(6): 1077-1093)

[2] Jacobson V, Smetters D K, Thornton J D, et al. Networking named content//Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. Rome, Italy, 2009: 1-12

[3] Smetters D, Jacobson V. Securing network content. Technical Report, PARC, TR-2009-1, CA, USA, 2009

[4] Koponen T, Chawla M, Chun B G, et al. A data-oriented (and beyond) network architecture. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 181-192

[5] Ahlgren B, Dannewitz C, Imbrenda C, et al. A survey of information-centric networking. IEEE Communications Magazine, 2012, 50(7): 26-36

[6] Arianfar S, Nikander P, Ott J. On content-centric router design and implications//Proceedings of the Re-Architecting the Internet Workshop. New York, USA, 2010: 5-10

[7] Arianfar S, Koponen T, Raghavan B, et al. On preserving privacy in content-oriented networks//Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking. Toronto, Canada, 2011: 19-24

[8] Rozhnova N, Fdida S. An effective hop-by-hop Interest shaping mechanism for CCN communications//Proceedings of the IEEE Conference on Computer Communications Workshops. Sedona, AZ, USA, 2012: 322-327

[9] Psaras I, Clegg R G, Landa R, et al. Modelling and evaluation of CCN-caching trees///Proceedings of the NETWORKING 2011. Springer Berlin Heidelberg, Valencia, Spain, 2011: 78-91

[10] Carofiglio G, Gallo M, Muscariello L, et al. Modeling data transfer in content-centric networking//Proceedings of the 23rd International Teletraffic Congress. International Teletraffic Congress, San Francisco, USA, 2011: 111-118

[11] Muscariello L, Carofiglio G, Gallo M. Bandwidth and storage sharing performance in information centric networking//Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking. Toronto, Canada, 2011: 26-31

[12] Carofiglio G, Gehlen V, Perino D. Experimental evaluation of memory management in content-centric networking//Proceedings of the 2001 IEEE International Conference on Communications (ICC). Kyoto, Japan, 2011: 1-6

[13] Rossi D, Rossini G. On sizing CCN content stores by exploiting topological information//Proceedings of the INFOCOM Workshops. Orlando, FL, USA, 2012: 280-285

[14] Wang S, Bi J, Wu J, et al. Could in-network caching benefit information-centric networking?// Proceedings of the 7th Asian Internet Engineering Conference. Bangkok, Thailand, 2011: 112-115

[15] Ming Z, Xu M, Wang D. Age-based cooperative caching in information-centric networks//Proceedings of the 31st Annual IEEE International Conference on Computer Communications. Florida, USA, 2012: 268-273

[16] Chai W K, He D, Psaras I, et al. Cache "less for more" in information-centric networks (extended version). Computer Communications, 2013, 36(7): 758-770

[17] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks//Proceedings of the 2nd Edition of the ICN Workshop on Information-Centric Networking. Orlando, USA, 2012: 55-60

[18] Li Z, Simon G. Time-shifted TV in content centric networks: The case for cooperative in-network caching//Proceedings of the IEEE International Conference on Communications (ICC). Kyoto, Japan, 2011: 1-6

[19] Cho K, Lee M, Park K, et al. WAVE: Popularity-based and collaborative in-network caching for content-oriented networks //Proceedings of the 31st Annual IEEE Conference on Computer Communications. Orlando, USA, 2012: 316-321

[20] Podlipnig S, Böszörmenyi L. A survey of web cache replacement strategies. ACM Computing Surveys (CSUR), 2003, 35(4): 374-398

WANG Guo-Qing, born in 1986, Ph. D. candidate. Her research interests include resource allocation and content-centric networking.

HUANG Tao, born in 1980, Ph. D., associate professor. His research interests include routing and forwarding, content-centric networking.

LIU Jiang, born in 1983, Ph. D., lecturer. His research interests include network virtualization, content-centric networking.

CHEN Jian-Ya, born in 1953, professor. His research interests include wireless access network and next generation internet.

LIU Yun-Jie, born in 1943, professor, Ph. D. supervisor, Member of Chinese Academy of Engineering. His research interests include next generation internet and network integration.

Background

As digital information diffuses over the Internet with an exponential growth, Internet architecture based on the end-point appears to be unsuited to deal with the trends. In order to solve the problem about the explosion of traffic, the concept of information-centric is proposed. Although the information-centric theory is realized by different architectures, the crucial issue of in-network caching for the popular contents is common to all. In-network storage for caching information objects is a fundamental component of information-centric approaches to networking, which can be leveraged to improve network performance significantly in terms of improving utilization and reducing propagation delay.

For the cache policy problem, the cache replacement policies have received much more attention than the cache decision policies. Most of the existing cache decision policies are either too complex to implemented in CCN, or have poor performance for the in-network caching. As a result, only a few cache decision policies can be used in CCN: Leave Copy Down (LCD), Random and ALWAYS. An obvious drawback of all of the three polices is that they make decision without considering the following factors directly: the content popularity, cache size and so on.

As every routers in CCN can cache contents, and Autonomous Systems (AS) will tend to fetch new contents from external ASs as less as possible, cooperative caching must be the most promising strategy in CCN, because it can increase the content diversity of caches within an AS. Even the cooperation needs some bandwidth transmitting between different caches (or routers). By considering the request rate and cache size, we design a new decision policy called ST to achieve cooperation among the caches. Through the ST-LRU policy, we can achieve the following advantages: (1) increase the hit rate; (2) reduce the copy times.

This work is supported by the National Basic Research Program (973 Program) of China (No. 2012CB315801), the National Natural Science Foundation of China (Nos. 61300184, 61302089) and the Fundamental Research Funds for the Central Universities (No. 2013RC0113). These projects aim to propose a new network architecture. Our group has been working on optimize the performance of the next generation Internet. Many high quality have been published in respectable journals, such as JZUS-C and so on.