基于内容分块流行度分级的信息中心网络缓存策略

曾宇晶,靳明双,罗洪斌
(北京交通大学下一代互联网互连设备国家工程实验室,北京 100044)

摘 要: 信息中心网络(ICN:Information-Centric Networking)域内缓存机制的研究大多假定单个内容的流行度相同,而忽视了同一内容的不同分块具有不同流行度的特性.本文提出了一种基于内容分块流行度以及缓存节点位置的分级缓存策略,通过兴趣包和数据包携带标签的方式实现隐式缓存协作.仿真实验证明相比于其他方案,该方案可以充分利用细粒度的内容分块流行度这一特性,提高缓存路由器缓存命中率,减小用户请求内容时延以及网络流量,进而提升用户对实时业务的服务体验.

关键词: 内容为中心网络;域内缓存;流行度
中图分类号: TN915 **文献标识码:** A **文章编号:** 0372-2112 (2016)02-0358-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.02.017

LICA:A Segment-Popularity Based Caching Scheme in ICN

ZENG Yu-jing,JIN Ming-shuang,LUO Hong-bin
(*National Engineering Lab for Next Generation Internet Interconnection Devices,Beijing Jiaotong University,Beijing* 100044,*China*)

Abstract: In-network caching schemes in information-centric networking (ICN) often consider the popularity granularity in the content level,which ignores the character that different parts of a content may have different popularity. This paper proposes a segment Level-based Implicit Cooperation cAche (LICA) scheme. LICA assigns a cache router a level based on its location and archives implicit cooperation with tags in interest and data packets. We compare LICA with other caching schemes and the simulation results show that LICA has higher average hit ratio,lower average hop count of content retrieval,and reduces the network traffic than other schemes,which will improve the experience of users in real-time streaming media service.

Key words: information centric networking;in-network caching;segment popularity

1 引言

随着互联网的高速发展,当前基于终端间包交换的点到点连接通信的TCP/IP网络模型已经无法适应用户对实时视频、流媒体等大流量数据业务越来越高的需求.为了解决互联网在传输、处理这些多媒体数据时效率低下,用户体验差,难以实现网络资源的高效利用等诸多问题,在未来网络体系的研究中不断有新的网络架构提出.其中ICN[1~3,17~19](Information-Centric Networking)架构脱颖而出,逐渐成为未来网络架构的研究热点.

与传统互联网端到端通信模式不同,ICN更强调以内容资源本身为中心.一方面,ICN通过对内容命名实现对内容的唯一标识,也实现了内容与其存储位置的分离.另一方面,ICN允许网络节点对内容进行缓存,这样可以使内容更接近用户,用户无需关心内容存储的位置,请求的消息将会被路由到临近提供该内容的节点并取回内容,从而减少服务器负载,降低网络带宽消耗以及提升用户体验.与此同时,内容也可以被分为具有独立名字的分块,进而在不同的节点缓存,同一内容的不同分块也会由于处于内容中的不同位置具有不同的流行度.

域内缓存是ICN架构中的重要组成部分,有效的部署域内缓存协作以及缓存替换策略对ICN网络的性能具有较大的影响.当前网络缓存协作的研究大多基于单个内容的流行度部署缓存,使最流行的内容能够缓存到离用户最近的缓存节点中,很少考虑到更细粒度的内容分块各自的流行度;另一方面,协作式缓存还存在缓存通告开销大,缓存信息同步慢导致兴趣请求包扑空等情况,从而降低了整个缓存系统的效率.

为了解决上述网络缓存协作中存在的问题,本文提出了LICA(segment Level-based Implicit Cooperation

收稿日期:2015-03-18;修回日期:2015-08-19;责任编辑:郭游
基金项目:国家973重点基础研究发展计划(No.2013CB329100);国家自然科学基金重点项目(No.61271200,No.61422101,No.61232017)

cAche)模型. LICA 旨在通过充分考虑内容分块粒度下的流行度,通过隐式协作的方式提高网络缓存效率,同时减小缓存协作开销. 本文的主要贡献如下:

(1)提出了基于内容分块流行度以及缓存节点位置的分级模型. (2)通过兴趣包和数据包携带标签的方式实现隐式协作. (3)通过大量的仿真实验,证明了 LICA 的优越性.

2　相关工作

传统单纯用来转发和路由的网络节点(如交换机,路由器)在 ICN 中被赋予了更多的作用,如对经过的内容具有缓存的能力. 在传统 WEB 缓存[4]以及代理缓存[5]研究的基础上,越来越多的基于 ICN 架构的域内缓存方案[6]被提出,用以提高网络缓存的效率进而提高整个网络的性能.

当前域内缓存协作机制可以根据协作的类型分为显式协作以及隐式协作两类. 显式协作通过节点与一定范围内的邻居节点同步各自的缓存信息来实现缓存的决策以及替换规则,因而显示协作往往伴随着大量缓存同步信息的通信开销,如文献[7]和文献[8]. 隐式协作则不需要缓存同步开销就能实现缓存协作. 文献[4]提出了 LCD 的策略,用户的单次请求将会使内容沿着路径下移一跳,这样请求次数越多的内容则会靠用户越近. ProbCache[9], WAVE[10], CLS[11]等都是采用与 LCD 相同的路径缓存的方式来进行缓存协作,只是具体协作的细节有所不同. 在文献[12]中,作者提出了一种基于时间权重的缓存协作方式,通过赋予离用户近以及请求次数高的内容高的时间权重来缓存更长的时间. CRCache[13]提出了 Router Importance(RI)的概念,将最流行的内容分块缓存到 RI 值最大的路由器上来提高整个网络的缓存效率,但是这样可能会造成最流行的内容无法缓存到离用户最近的位置. TLRU[14]对 Least Recently Used(LRU)策略进行了修改,对内容大小以及被请求次数的统计,提出了基于时间认知的 LRU 方案,但并没有在全局网络形成很好的协作因而对缓存性能的提高有限. IPEC[15]考虑到了同一内容的不同分块具有不同的流行度,并提出了与 WAVE 类似的路径缓存协作方案,但是由于分块粒度的不够因而无法充分利用分块流行度的这一特性.

为了更好的利用细粒度的内容分块流行度这一特性,我们提出了通过统计用户的请求模式来对内容分块进行分级的 ICN 域内缓存协作方案.

3　基于内容分块流行度分级的隐式协作缓存(LICA)模型

本文提出了一种基于内容分块流行度分级的隐式协作缓存 Level-based Implicit Cooperation cAche(LICA)模型,具有以下特点:

(1)去中心化,即不需要专门的网络设备来收集并统计网络中内容的流行度并控制各缓存路由器中内容的副本的分布以及替换,只需要由离用户最近的路由器统计用户请求信息即可.

(2)内容分块流行度分级,即根据缓存路由器空间大小将不同流行度的内容分块分为不同的等级,等级高的内容分块会被缓存到离用户近的路由器,提高网络缓存效率.

(3)隐式协作,即缓存路由器之间不需要通过额外的缓存消息进行通告,也不需要在本地维护相应的邻居缓存列表,缓存标志仅由原有的请求包与数据包携带,减少额外通信开销.

3.1　LICA 概述

LICA 方案可以在不同的 ICN 网络架构下部署. 为了描述方便,我们以 ICN 网络中比较有代表性的 CCN(Content Centric Networking)架构[1]为例对 LICA 方案进行描述. 在一个 CCN 网络中,该网络由 1 个内容源服务器,N 个缓存路由器以及 M 个用户组成. 内容源服务器存储有网络中所有的内容数据,内容数据可以被分块. 假设所有分块大小相同且为单位数量 1,每个缓存路由器都有相同大小的缓存空间 S,即可以同时缓存内容分块的数量为 S.

根据传统 CCN 网络的工作机制,内容数据可以被分块,当用户想要获取某个内容分块时,用户将向最近的缓存路由器发送包含该分块名称的兴趣包. 如果该内容包含 10 个分块,用户想要获取整个内容的数据,则需要发送 10 个兴趣包. 关于内容合适的分块数量以及分块数量带来的可扩展性问题,不在本文的讨论范围中,但本方案对粗粒度(单个内容),细粒度(分包)都能够提供较好的支持. 如果缓存路由器存储了该内容分块则会返回分块给用户,没有则会转发到下一跳路由器. 如果沿途路由器都没有缓存该分块,兴趣包将会被转发到内容源服务器来满足服务请求. 内容分块的数据包将沿着之前兴趣传输路径反向传给用户,沿途经过的缓存路由器将会根据本地缓存策略决定是否缓存该分块.

与 CCN 不同,我们修改了兴趣包与数据包的格式,加入了分块等级(如何确定分块等级,请见 3.3.1 节)的字段,通过携带和传递分块的等级来决定分块是否被缓存路由器缓存. 与此同时,根据缓存路由器距离用户的跳数,我们将缓存路由器分为不同的层级. 另外,为了统计用户请求信息,我们还在第一跳缓存路由器加入了用户请求统计列表 Interest Statistic Table(IST). IST 根据用户请求统计以及缓存路由器缓存空间对内容分

块进行分级,请求频率高的分块对应高的分块等级. 当兴趣包到达后,第一跳缓存路由器在查询 IST 后填充兴趣包分块等级的字段. 当兴趣包被沿途某个缓存路由器或者最终的内容源服务器满足时,兴趣包中的分块等级字段将会被读取并写入到内容分块的数据包中,用来决定数据包是否被缓存.

3.2 LICA 工作原理

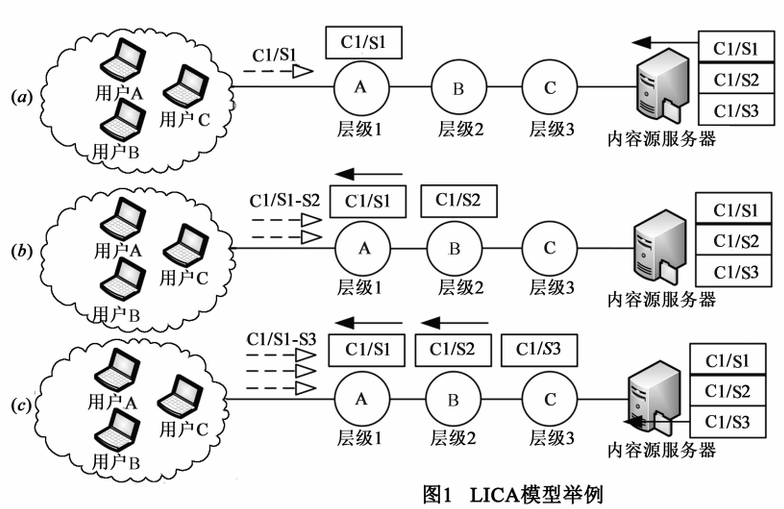
下面阐述 LICA 的工作原理. 如图 1 所示,根据距离用户的跳数,缓存路由器 A,B,C 被分为层级 1 – 3. 同时假设每个路由器的缓存空间为 1,即只能缓存一个缓存分块. 内容 C1 被分为 3 个大小相同的分块 S1,S2,S3.

兴趣包中分块等级初始值为 0. 在图 1(a)中,用户 C1 发送兴趣包请求内容 C1 的分块 S1. 兴趣包到达第一跳缓存路由器 A 后,A 在 IST 中加入请求的内容分块名称 C1/S1,已请求的次数 1,以及经过计算之后的等级 1 并在兴趣包中的等级字段写入 1. 由于沿路路由器都没有缓存该分块,兴趣包将会被路由到内容源服务器来满足请求,内容源服务器读取兴趣包中的等级字段并且将等级 1 写入到内容分块的数据包中,沿原路径返回到用户.

沿途缓存路由器将会检查数据包中的等级字段,以判断是否与自己的层级数相等,相等则缓存,不相等则忽略. 最终,内容分块 C1/S1 被层级 1 的缓存路由器 A 缓存.

接下来,如图 1(b)所示. 用户 B 先后发出两个兴趣包分别请求分块 C1/S1 与 C1/S2. 当 C1/S1 的兴趣包到达 A 后,首先查询到 IST 表中对应的字段,添加次数并重新计算等级. 然后查询本地缓存发现已经存取了 C1/S1 分块,将等级赋值到分块数据包后,直接发送分块给用户;当 C1/S2 的兴趣包到达 A 后,由于每个缓存路由器只能缓存一个分块且 C1/S2 在 IST 表中请求的次数少于 C1/S1,因此分块等级为 2. A 在 IST 中加入请求的内容分块名称 C1/S2,已请求的次数 1,以及经过计算之后的等级 2 并在兴趣包中的等级字段写入 2. 之后流程与图 1(a)中相似,最终 C1/S2 被层级 2 的缓存路由器 B 缓存.

在图 1(c)中,用户 C 想要获取整个内容 C1,先后发送了三个兴趣请求包请求 S1,S2,S3. A 更新 IST 表后,S1 由 A 满足请求直接返回分块. S2 由 B 满足请求返回分块. S3 的兴趣包从 IST 表获得的等级值为 3,因此最终分块 S3 被缓存在 C 处.



节点A的IST表

名字	次数	等级
C1/S1	1	1

节点A的IST表

名字	次数	等级
C1/S1	2	1
C1/S2	1	2

节点A的IST表

名字	次数	等级
C1/S1	3	1
C1/S2	2	2
C1/S2	1	3

图1 LICA模型举例

在图 1 的例子中我们可以看到,分块所处缓存路由器的层级与其流行度对应,被请求次数越多的分块将会被缓存到离用户越近的缓存路由器上. 同时,缓存分块是否缓存以及缓存的位置直接通过第一跳路由器就能决定,不需要其他复杂的通信以及协作,这样既提高了整个网络的缓存效率也避免了不必要的开销.

3.3 LICA 特性详述

前文中我们对 LICA 的工作原理进行了描述,接下来我们将详细介绍 LICA 中缓存路由器层级判断方法、IST 表的更新方法以及缓存更新替换方法.

3.3.1 缓存路由器层级

缓存路由器根据收到的兴趣包中所记录的跳数来确定自身的层级. 当缓存路由器收到兴趣包后,检查兴趣包中跳数的 H_i,并与自身层级 H_R 做比较,当跳数字段 H_i 大于自身层级 H_R 时,将当前层级 H_R 修改为跳数 H_i 所对应的数值. 当跳数字段 H_i 小于等于自身层级 H_R 时,则不需要修改自身层级 H_R. 在一个信息中心网络域内,缓存路由器数量一定,因而缓存路由器层级也是不变的,且每个缓存路由器都有自己确定的层级.

3.3.2 IST 表

IST 表只会在第一跳缓存路由器生成,其他层级的路由器不会有 IST 表,这样可以有效减少表的存储开销以及更新开销. 但是当某个缓存路由器直接连接用户

且同时位于其他路径的非第一跳位置时,该缓存路由器同样生成 IST 表. 这样做的目的是确保所有的兴趣包都能被分配等级. IST 表格式如图 1 所示,包括分块名字,被请求次数和等级. 其中分块名字由分块所属内容名字与分块处于内容中分块位置的编号构成,它是唯一的并可以用来区别其他分块.

整个 IST 表按照被请求次数项排序. 被请求次数项记录了一段时间内该分块被请求的次数. 这里所说的一段时间可以是一天,一个星期或者一个月,根据总的请求次数来进行调节和统计. 同时,我们也设置了被请求次数的最大阈值 R_T. 当表中最多次数项的数值达到最大阈值 R_T 后,表中所有项的次数都会被减半后向下取整. 这样做的目的有:(1)减少存在历史数据影响将来数据的"缓存污染"效用,那些近段时间内被请求的次数多的内容分块可以在较短的时间内获得比较高的等级而不受那些近段时间内被请求次数少的内容分块的影响;(2)剔除那些被请求次数为 1 的内容分块,从而减少整个 IST 表的大小.

IST 表中最重要的表项是等级项,分块的等级决定了分块被缓存的位置. 假设分块请求次数项排序数为 R_S,则分块的等级 L_S 与 R_S 的关系如下:

$$L_S = \lceil R_S / S \rceil \tag{1}$$

其中 S 为缓存路由器可以容纳的分块数. 由式(1)可以得出,每个等级上分块的数量即为缓存路由器可以容纳的分块数量. 因此当分块在请求次数项的排序发生变化时,其分块等级并不一定会发生变化,只有当位于等级边缘的两个分块发生排序更迭时才会出现分块等级的变化,进而引发缓存路由器的缓存更新替换. 这样的策略有效的减少了分块在沿途缓存路由器的移动,也减少了缓存路由器对分块重复存储的开销.

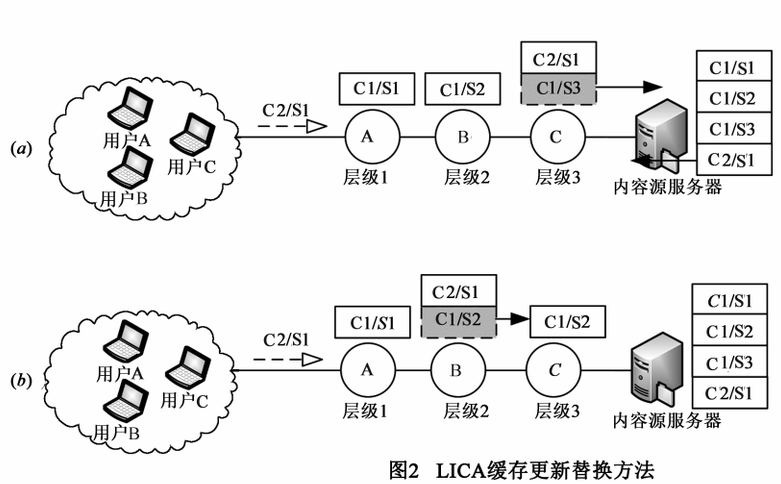
3.3.3　缓存更新替换

下面介绍当分块等级发生变化时,如何进行缓存的更新与替换.

我们依旧采用章节 3.2 中的例子来进行说明,如图 2(*a*)所示. 用户 A 再次发送兴趣包,请求另一个内容分块 C2/S1. 缓存路由器在接收到兴趣包后添加一个C2/S1的条目到 IST 表中. 相同请求次数的情况下,新添加的条目将具有更高的排序,因此 C2/S1 的等级将会比具有相同请求次数的 C1/S3 高. 所以当内容源服务器将具有等级 3 的内容分块 C2/S1 返回到缓存路由器 C 时,C 首先将已存取的内容分块 C1/S3 分块等级加 1,然后剔除并向上游路由器发送,进而释放出空间用来缓存 C2/S1. 由于 C 的上游是内容源服务器,因此 C1/S3 将直接被丢弃. 此后 C2/S1 沿路径到达用户 A 满足请求.

在图 2(*b*)中,用户 B 再次发出兴趣包请求分块 C2/S1. 当兴趣包到达 A 后,A 更新 IST 表,C2/S1 的被请求次数变为 2 从而超越分块 C1/S2,两者的分块等级也发生了互换. 当兴趣包到达 C 后,请求将直接由 C 处的缓存满足. 当分块 C2/S1 到达 B 后,原本缓存在 B 的 C1/S2 会被修改分块等级推送到上层路由器 C 进行缓存. 然后 B 缓存分块 C2/S1.

当缓存路由器本地缓存有兴趣包所请求的分块,缓存路由器将兴趣包中携带的分块等级与本地缓存分块等级进行比较:(1)兴趣包分块等级高于本地缓存分块等级,缓存分块将会在本地删除并向下游路由器传输满足请求并缓存;(2)兴趣包分块等级等于本地缓存分块等级,缓存分块将不会在本地删除并向下游路由器传输满足请求;(3)兴趣包分块等级低于本地缓存分块等级,缓存分块将会在本地删除并同时向下游路由器传输满足请求向上游路由器传输进行缓存.



节点A的IST表

名字	次数	等级
C1/S1	3	1
C1/S2	2	2
C2/S1	1	3
C1/S3	4	4

节点A的IST表

名字	次数	等级
C1/S1	3	1
C1/S2	2	2
C2/S1	1	3
C1/S3	4	4

图2　LICA缓存更新替换方法

同时缓存路由器采用 LRU 算法来进行缓存替换. 当某个缓存路由器缓存空间满了后,再有新的内容分块到达需要缓存时,缓存路由器会把缓存队列中最久没有被请求的缓存分块移除,再把其分块等级加 1 后发送给上游缓存路由器. 由于下游的缓存路由器存储的内容分块比上游存储的内容分块具有更高的流行度,因此将移除的缓存分块推送给上游路由器而不是简单的删除可以有效的减少再次请求并缓存的过程,同时也能提高缓存命中率.

这样单纯通过兴趣包与内容分块数据包中分块等级的携带,就能实现缓存分块按照用户请求的流行度进行分布,节省了缓存路由器之间缓存信息的通告开销,也不需要收集全网的内容分布信息来进行缓存决策,最大化的利用了缓存资源,实现了缓存路由器之间路由缓存的隐式协作.

3.3.4 LICA 方案定性分析

与其他隐式协作方案相比[9~11],本方案在兴趣包与内容分块数据包原包的基础上加入分块等级的字段来实现缓存路由器之间的缓存隐式协作,因而增加了每个缓存路由器读取该字段的开销以及与本身层级比较的计算代价. 但相比于其他显式协作方案[7,8],本方案在实现缓存协作的过程中并没有引入新的数据包,只需要读取一个字段的数据即可以实现缓存路由器之间的协作. 相比于缓存空间利用率以及缓存效率的提高,这些代价可以忽略不计.

另一方面,本方案需要每个连接用户的缓存路由器中维护一个 IST 表,并对收到请求进行统计并划分等级,相当于维护一个 LFU 表并对数量为 n 的数据进行查找,添加和删除,故每次操作的时间复杂度为 $O(n)$. 但由于缓存节点层级有限,且每个缓存节点缓存空间有限,缓存节点可以根据缓存节点层级和缓存空间的大小决定 IST 表的阈值,当表项超过该阈值时对表项进行清理,进而可以减小维护 LST 表的复杂度. 同时,可以通过引入最小堆以及 hash 表等方法来实现对 LFU 算法的优化[20],进而降低 LST 表的维护开销.

4 仿真分析

为了验证本文提出的 LICA 策略的缓存性能,本文在 OMNeT++ 平台上开发了模拟器对以上方案进行仿真实现以及性能评估. 主要考虑的性能参数包括路由器缓存命中率,用户获取内容跳数值以及源服务器流量与系统总流量的比值.

4.1 仿真环境和参数配置

我们将 LICA 与 WAVE、ProbCache 以及 CLS 方案进行了对比. 其中我们假设 WAVE 的参数与[10]中相同且 $x=2$,ProbCache 的沿路缓存概率为 0.7,CLS 的参数设置为 Hth = 2. 仿真拓扑采用与 ProbCache 以及 CLS 相同的树形拓扑. 拓扑中包含 5 个层级 31 个路由器,其中每 1000 个用户与底端 16 个叶子节点的路由器相连,源服务器与顶端的根节点的路由器相连,如图 3 所示. 同时,我们假设内容数量为 10000 个,每个内容大小相同为 10Mbytes,并且平均分为 k = 10 个分块;每个缓存路由器的缓存空间为 1000Mbytes,即可以缓存 1000 个缓存分块. 此外,在仿真过程中每个用户将会向连接的路由器发送 10000 个对内容的请求,并且请求的到达符合泊松分布. 用户对内容的请求分布符合 Zipf 分布[16],其中 $0.5\leqslant\alpha\leqslant1$,默认情况下 $\alpha=0.8$.

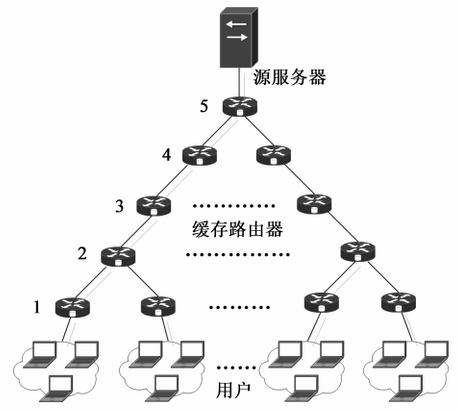


图3 二叉树型拓扑

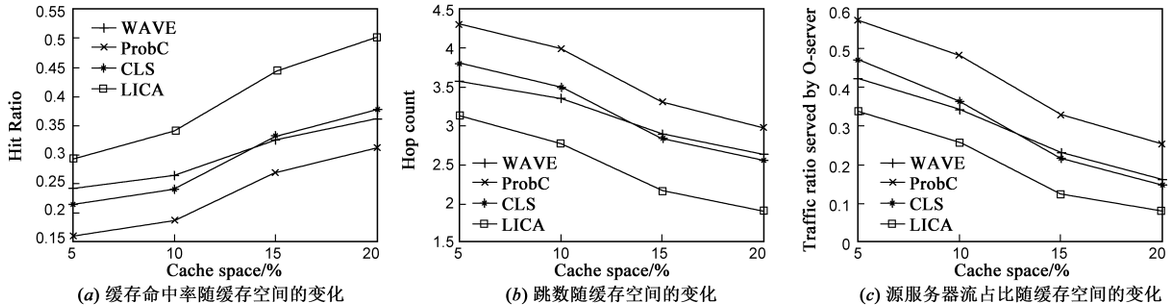
4.2 仿真结果分析

首先我们考虑缓存空间大小对系统缓存性能的影响. 每个缓存路由器的缓存空间 S 分别设置成 500 Mbytes、1000 Mbytes、1500 Mbytes 和 2000Mbytes. 如图 4 所示,我们考虑不同缓存空间下四种缓存策略的内容分块平均缓存命中率、请求的平均跳数与源服务器流量占比的仿真对比. 如图 4(a)所示,随着路由器缓存空间的变化,LICA 比其他三种缓存路由器具有更高的缓存命中率,并且增长速度也比其他的策略要高. LICA 的缓存命中率在不同 S 时比 CLS 分别高 37.4%,41.4%,33.5% 以及 33.2%. 如图 4(b)所示,LICA 在请求平均跳数这项能比较中也具有更好的性能. 其中当 S = 2000Mbytes 时,LICA 的请求评价跳数为 1.89,比其他三种缓存策略分别小 0.74,1.09 以及 0.65 跳. 图 4(c)则展示了源服务器流量占比受 α 的影响,源服务器流量占比表示源服务器发送的内容流量与全网所有发送的内容流量和的比值,源服务器流量占比从网络流量的角度反映了缓存系统的性能. 当 S = 2000Mbytes 时,LICA 的源服务器流量占比仅为 7.8%,说明绝大部分的流量都又缓存路由器提供,大大减小了系统流量开销.

接下来我们考虑 Zipf 分布参数 α 对系统缓存性能

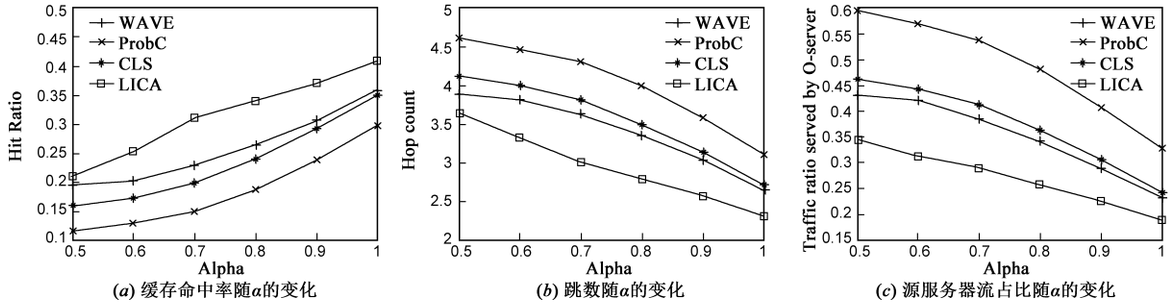
的影响. 图 5 是 LICA 与 WAVE、ProbCache 以及 CLS 四种缓存策略在 Zipf 分布参数 α 变化情况下,内容分块平均缓存命中率、请求的平均跳数与源服务器流量占比的仿真对比. 如图 5(a)所示,当 $0.5 \leqslant \alpha \leqslant 1$,通过统计拓扑中所有路由器的缓存命中率,计算整个系统的内容分块平均缓存命中率. 在 α 变化的过程中,LICA 始终比其他三种缓存策略具有更高的缓存命中率,其中当 $\alpha = 0.8$ 时,LICA 的缓存命中率比 WAVE、ProbCache 以及 CLS 分别提高了 28.8%、82.1% 和 41.4%. 如图 5(b)所示,LICA 在请求平均跳数这项性能中也展示了很大的优势. 当 $\alpha = 0.8$ 时,LICA 的请求平均跳数比其他三种策略分别提高了 0.55、1.21 和 0.71 跳. 通过 LICA 的方案,用户可以从更近的缓存路由器获得内容,从而减少了用户等待时延,提高了用户体验. 图 5(c)则展示了源服务器流量占比受 α 的影响,相比与其他三个缓存策略,当 α 变化时 LICA 有更小的源服务器流量占比. 当 $\alpha = 0.8$ 时,LICA 的源服务器流量占比分别为 WAVE、ProbCache 和 CLS 的 75.5%、53.6% 和 71.9%.

综上所述,本文提出的 LICA 缓存策略在内容分块平均缓存命中率、请求的平均跳数与源服务器流量占比这三种缓存性能参数中都比其他三种缓存策略有更好的缓存性能.



(a) 缓存命中率随缓存空间的变化　(b) 跳数随缓存空间的变化　(c) 源服务器流占比随缓存空间的变化

图4　缓存空间大小对系统缓存性能影响



(a) 缓存命中率随α的变化　(b) 跳数随α的变化　(c) 源服务器流占比随α的变化

图5　Zipf分布参数α对系统缓存性能影响

5　结束语

本文针对现有 ICN 网络域内缓存策略没有充分考虑到内容分块请求流行度的问题,提出了一种基于内容分块流行度分级的缓存策略. 仿真测试结果表明:该方案可以利用细粒度的内容分块流行度这一特性,提高缓存路由器缓存命中率,减小用户请求内容时延进而提升用户对实时业务的服务体验,又可减小整个系统的流量. 下一步工作将研究大规模系统下集中控制缓存策略的方案设计.

参考文献

[1] Zhang L X, Estrin D, Burke J, et al. Named Data Networking (NDN) project[R]. California: PARC, 2010.

[2] Koponen T, Chawla M, Chun B. G, et al. A data-oriented (and beyond) network architecture[J]. SIGCOMM Comput Commun Rev, 2007, 37(4): 181 – 192.

[3] Lagutin D, Visala D, Tarkoma S. Publish/subscribe for internet: Psirp perspective[R]. Towards the Future Internet Emerging Trends from European Research, 2010(4): 75 – 84.

[4] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical web caches[A]. Proceedings of 2004 IEEE International Conference on Performance, Computing, and Communications[C]. Phoenix, 2004. 445 – 452.

[5] Chae Y, Guo K, Buddhikot M M, et al. Silo, rainbow, and caching token: Schemes for scalable, fault tolerant stream caching[J]. Selected Areas in Communications, IEEE Journal on, 2002, 20(7): 1328 – 1344.

[6] Zhang G, Li Y, Lin T. Caching in information centric networking: a survey[J]. Computer Networks, 2013, 57(16):

3128 – 3141.
[7] Long Y, Fan L, Yan Z. Off-path and on-path collaborative caching in named data network[A]. Proceedings of AsiaFI[C]. Kyoto, 2012. 1 – 8.
[8] Vassilakis V G, Al-Naday M F, Reed M J, et al. A cache-aware routing scheme for information-centric networks[A]. Proceedings of 2014 9th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)[C]. Manchester, 2014: 721 – 726.
[9] Psaras I, Chai, Wei Koong, Pavlou G. Probabilistic in-network caching for information-centric networks[A]. Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking[C]. Helsinki, 2012. 55 – 60.
[10] Cho K, Lee M, Park K, et al. Wave: Popularity-based and collaborative in-network caching for content-oriented networks[A]. Proceedings of 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)[C]. Orlando, 2012. 316 – 321.
[11] Li Yang, Lin Tao, Tang Hui, et al. A chunk caching location and searching scheme in content centric networking[A]. Proceeding of 2012 IEEE International Conference on Communications (ICC)[C]. Ottawa, 2012. 2655 – 2659.
[12] Ming Zhong-xing, Xu Ming-wei; Wang Dan. Age-based cooperative caching in Information-Centric Networks[A]. Proceedings of 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)[C]. Orlando, 2012. 268 – 273.
[13] Wang Wei, Sun Yi, Guo Yang, et al. CRCache: Exploiting the correlation between content popularity and network topology information for ICN caching[A]. Proceeding of 2014 IEEE International Conference on Communications (ICC)[C]. Sydney, 2014. 3191 – 3196.
[14] Bilal M, Kang S G. Time Aware Least Recent Used (TLRU) cache management policy in ICN[A]. Proceeding of 2014 16th International Conference on Advanced Communication Technology (ICACT)[C]. PyeongChang Korea, 2014. 528 – 532.
[15] Lim Sung-Hwa, Ko Young-Bae, Jung Gue-Hwan, et al. Inter-Chunk Popularity-Based Edge-First Caching in Content-Centric Networking[J]. IEEE Communications Letters, 2014, 18(8): 1331 – 1334.
[16] Breslau L, Cao P, Fan L, et al. Web caching and zipf-like distributions: evidence and implications[A]. Proceedings of INFOCOM'99[C]. New York, 1999. 126 – 134.
[17] Xylomenos G, Ververidis C, Siris V, et al. A Survey of Information-Centric Networking Research[J]. IEEE Communications Surveys and Turorials, 2014, 16(2): 1024 – 1049.
[18] Ahlgren B, Dannewitz C, Imbrenda C, et al. A Survey of Information-Centric Networking[J]. IEEE Communications Magazine, 2012, 50(7): 26 – 36.
[19] Luo H, Chen Z, Cui J, et al. CoLoR: an information-centric Internet architecture for innovations[J]. IEEE Network Magazine, 2014, 28(3): 4 – 10.
[20] Ketan S, Mitra A, Matani D. An $O(1)$ algorithm for implementing the LFU cache eviction scheme[OL]. http://dhruvbird.com/lfu.pdf, 2010.

作者简介

曾宇晶 男, 1987 年 8 月出生, 湖南邵东人, 博士研究生, 2009 年在北京交通大学获得工学学士学位, 并于同年开始攻读工学博士学位. 主要从事下一代互联网、智慧协同网络、信息中心网络的路由与缓存策略研究.
E-mail: 09111043@bjtu.edu.cn

靳明双 女, 1990 年 9 月出生, 黑龙江省哈尔滨人, 博士研究生, 2013 年在北京交通大学获得工学学士学位, 并于同年硕博连读攻读工学博士学位. 主要从事下一代互联网、智慧协同网络、软件定义网络方面的研究.
E-mail: 13120082@bjtu.edu.cn

罗洪斌 男, 1977 年 1 月出生, 重庆忠县人, 北京交通大学教授、博士生导师, 现任北京交通大学下一代互联网互连设备国家工程实验室副主任、中国通信学会第四届青年工作委员会委员、北京通信学会第八届理事会理事, 入选教育部新世纪优秀人才支持计划. 目前主要从事未来互联网体系架构、理论与关键技术的研究, 正主持多项国家 973、863、国家自然科学基金项目. 近年来在 IEEE/ACM Transactions on Networking, IEEE Journal on Selected Areas in Communications, IEEE Network 等本领域高水平期刊与国际会议发表论文 50 余篇.
E-mail: hbluo@bjtu.edu.cn